LAPP
Laboratoire d'Annecy de Physique des Particules

Retour GTC 2024 : GPU Technical Conference

Pierre Aubert

ESCAPE
cta
cherenkov telescope array
cnrs
UNIVERSITÉ SAVOIE MONT BLANC

Introduction first part

GTC 2024 Presentation Catalog

This is not possible to summarize **1080** talks !

Only focus on :

- **Hardwares** evolution from 2020 to 2023
- **Compilers** evolution
- **Feedback**

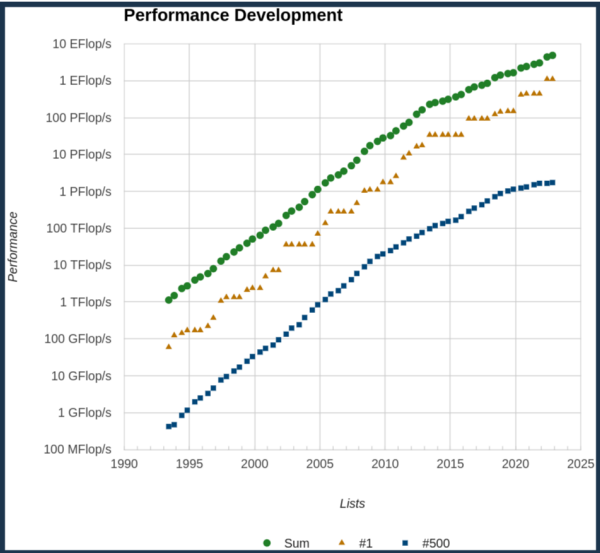
Keynotes video link

Top / Green 500

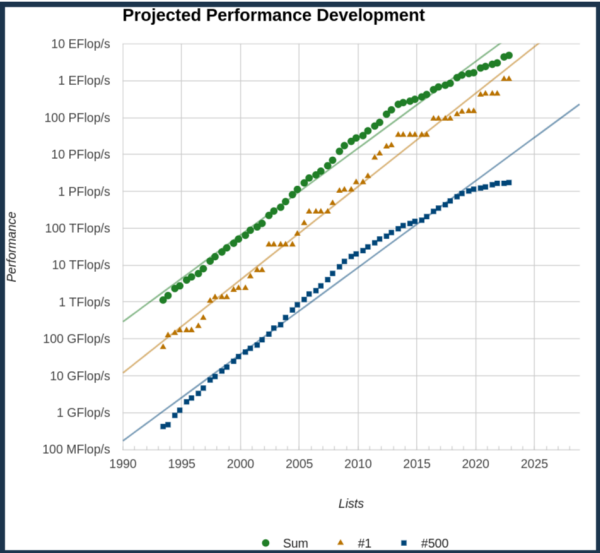
Top 500 :
- Top of the most **powerful** computing center in the world

Green 500 :
- Top of the most **efficient** computing center in the world

Top / Green 500

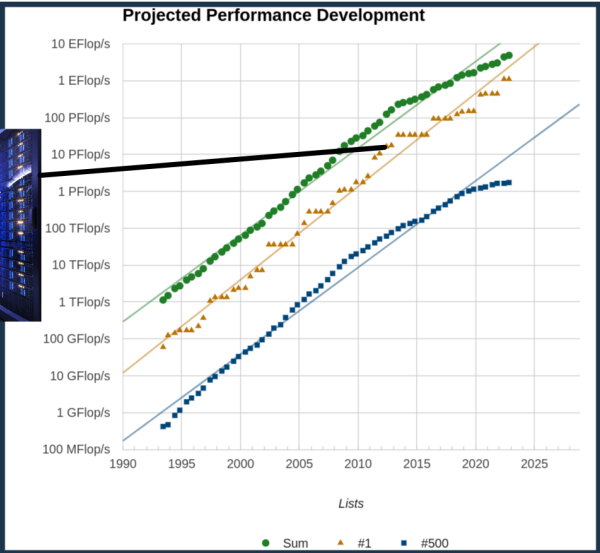


Top / Green 500



Top / Green 500

Titan (2012)
- **20 PFlops**
- **7 MW**



Top / Green 500

Top 500 (11/2022)

Rank	System	Cores	(PFlop/s)	(PFlop/s)	(kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, **Fujitsu** RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, **Atos** EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM** DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
6	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM / NVIDIA / Mellanox** DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438
7	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, **NRCPC** National Supercomputing Center in Wuxi China	10,649,600	93.01	125.44	15,371
8	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, **HPE** DOE/SC/LBNL/NERSC United States	761,856	70.87	93.75	2,589
9	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, **Nvidia** NVIDIA Corporation United States	555,520	63.46	79.22	2,646
10	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, **NUDT** National Super Computer Center in Guangzhou China	4,981,760	61.44	100.68	18,482

Top / Green 500

Top 500 (11/2022)

Rank	System	Cores	(PFlop/s)	(PFlop/s)	(kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100 **GPU**
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.20GHz, Tofu interconnect D, **Fujitsu** RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016 **GPU**
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.60GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, **Atos** EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610 **GPU**
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM** DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096 **GPU**
6	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM / NVIDIA / Mellanox** DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438 **GPU**
7	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, **NRCPC** National Supercomputing Center in Wuxi China	10,649,600	93.01	125.44	15,371
8	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, **HPE** DOE/SC/LBNL/NERSC United States	761,856	70.87	93.75	2,589 **GPU**
9	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, **Nvidia** NVIDIA Corporation United States	555,520	63.46	79.22	2,646 **GPU**
10	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, **NUDT** National Super Computer Center in Guangzhou China	4,981,760	61.44	100.68	18,482 **GPU**

Top / Green 500

Top 500 (11/2022)

Rank	System	Cores	(PFlop/s)	(PFlop/s)	(kW)	
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100	GPU
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, **Fujitsu** RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899	ARM CPU
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016	GPU
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, **Atos** EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610	GPU
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM** DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096	GPU
6	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM / NVIDIA / Mellanox** DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438	GPU
7	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, **NRCPC** National Supercomputing Center in Wuxi China	10,649,600	93.01	125.44	15,371	SW26010 CPU
8	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, **HPE** DOE/SC/LBNL/NERSC United States	761,856	70.87	93.75	2,589	GPU
9	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, **Nvidia** NVIDIA Corporation United States	555,520	63.46	79.22	2,646	GPU
10	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, **NUDT** National Super Computer Center in Guangzhou China	4,981,760	61.44	100.68	18,482	GPU

Top / Green 500

Top 500 (11/2022)

Rank	System	Cores	(PFlop/s)	(PFlop/s)	(kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
	GPU				
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, **Fujitsu** RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
	ARM CPU				
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
	GPU				
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, **Atos** EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610
	GPU				
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM** DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
	GPU				
6	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM / NVIDIA / Mellanox** DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438
	GPU				
7	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, **NRCPC** National Supercomputing Center in Wuxi China	10,649,600	93.01	125.44	15,371
	SW26010 CPU				
8	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, **HPE** DOE/SC/LBNL/NERSC United States	761,856	70.87	93.75	2,589
	GPU				
9	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, **Nvidia** NVIDIA Corporation United States	555,520	63.46	79.22	2,646
	GPU				
10	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, **NUDT** National Super Computer Center in Guangzhou China	4,981,760	61.44	100.68	18,482
	GPU				

Green 500 (11/2022)

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	Power (kW)	Energy Efficiency (GFlops/watts)
1	405	**Henri** - Lenovo ThinkSystem SR670 V2, Intel Xeon Platinum 8362 32C 2.80GHz, [3X]L NVIDIA H100 80GB PCIe, Infiniband HDR, **Lenovo** Flatiron Institute United States	5,920	2.04	31	65.091
2	32	**Frontier TDS** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	120,832	19.20	309	62.684
3	11	**Adastra** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** Grand Equipement National de Calcul Intensif - Centre Informatique National de l'Enseignement Superieur (GENCI-CINES) France	319,072	46.10	921	58.021
4	15	**Setonix - GPU** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** Pawsey Supercomputing Centre, Kensington, Western Australia Australia	181,248	27.16	477	56.983
5	68	**Dardel-GPU** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** KTH - Royal Institute of Technology Sweden	52,864	8.26	146	56.491
6	1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	21,100	52.227
7	3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** EuroHPC/CSC Finland	2,220,288	309.10	6,016	51.382
8	159	**ATOS THX.A.B** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, **Atos** Atos France	25,056	3.50	86	41.411
9	359	**MN-3** - MN-Core Server, Xeon Platinum 8260M 24C 2.40Hz, Preferred Networks MN-Core, MN-Core DirectConnect, **Preferred Networks** Preferred Networks Japan	1,664	2.18	53	40.901
10	331	**Champollion** - Apollo 6500, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 80 GB, Mellanox HDR Infiniband, **HPE** Hewlett Packard Enterprise France	19,840	2.32	60	38.555

Top / Green 500

Top 500 (11/2022)

Rank	System	Cores	(PFlop/s)	(PFlop/s)	(kW)	
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100	GPU
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, **Fujitsu** RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899	ARM CPU
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016	GPU
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.60Hz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, **Atos** EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610	GPU
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM** DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096	GPU
6	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, **IBM / NVIDIA / Mellanox** DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438	GPU
7	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, **NRCPC** National Supercomputing Center in Wuxi China	10,649,600	93.01	125.44	15,371	SW26010 CPU
8	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, **HPE** DOE/SC/LBNL/NERSC United States	761,856	70.87	93.75	2,589	GPU
9	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, **Nvidia** NVIDIA Corporation United States	555,520	63.46	79.22	2,646	GPU
10	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, **NUDT** National Super Computer Center in Guangzhou China	4,981,760	61.44	100.68	18,482	GPU

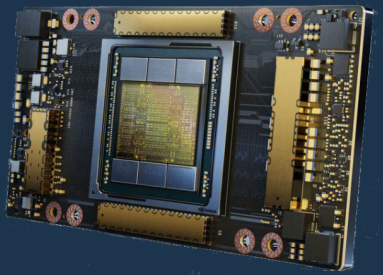
Green 500 (11/2022)

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	Power (kW)	Energy Efficiency (GFlops/watts)	
1	405	**Henri** - Lenovo ThinkSystem SR670 V2, Intel Xeon Platinum 8362 32C 2.80GHz (ISC), NVIDIA H100 80GB PCIe, Infiniband HDR, **Lenovo** Flatiron Institute United States	5,920	2.04	31	65.396	GPU
2	32	**Frontier TDS** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	120,832	19.20	309	62.684	GPU
3	11	**Adastra** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** Grand Equipement National de Calcul Intensif - Centre Informatique National de l'Enseignement Suprieur (GENCI-CINES) France	319,072	46.10	921	58.021	GPU
4	15	**Setonix - GPU** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** Pawsey Supercomputing Centre, Kensington, Western Australia Australia	181,248	27.16	477	56.983	GPU
5	68	**Dardel-GPU** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** KTH - Royal Institute of Technology Sweden	52,864	8.26	146	56.483	GPU
6	1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	21,100	52.227	GPU
7	3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, **HPE** EuroHPC/CSC Finland	2,220,288	309.10	6,016	51.382	GPU
8	159	**ATOS THI.A.B** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, **Atos** Atos France	25,056	3.50	86	41.43	GPU
9	359	**MN-3** - MN-Core Server, Xeon Platinum 8260M 24C 2.4GHz, Preferred Networks MN-Core, MN-Core DirectConnect, **Preferred Networks** Preferred Networks Japan	1,664	2.18	53	40.957	GPU
10	331	**Champollion** - Apollo 6500, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 80 GB, Mellanox HDR Infiniband, **HPE** Hewlett Packard Enterprise France	11,840	2.32	60	38.555	GPU

2020 - Ampere GPU

2020 - Ampere GPU

A100 (Ampere)

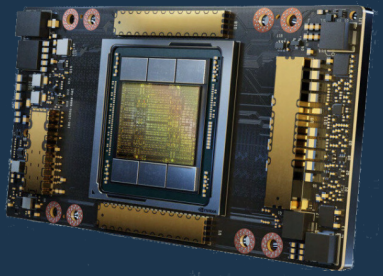


A100 SXM

2020 - Ampere GPU

7 nm

A100 (Ampere)



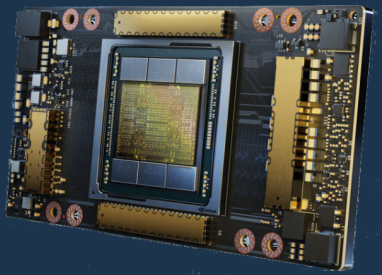
A100 SXM

2020 - Ampere GPU

7 nm

Compute Capabilites **8.0**

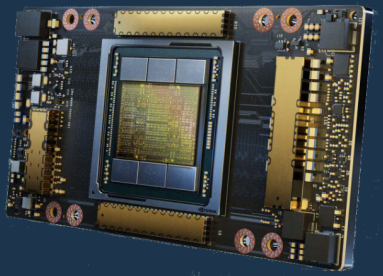
A100 (Ampere)



A100 SXM

2020 - Ampere GPU

A100 (Ampere)



A100 SXM

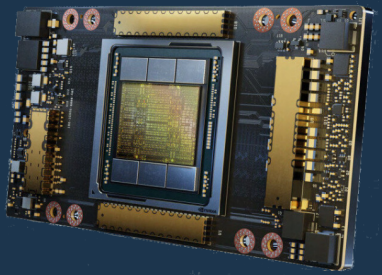
7 nm

Compute Capabilites **8.0**

108 SMs **6912** Cores

2020 - Ampere GPU

A100 (Ampere)



A100 SXM

7 nm

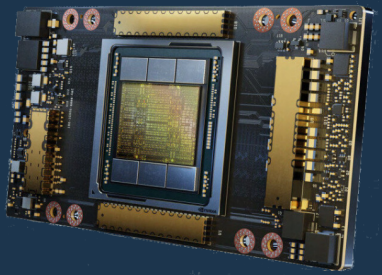
Compute Capabilites **8.0**

108 SMs **6912** Cores

Peak **19.5 TFlops** (without tensor cores)

2020 - Ampere GPU

A100 (Ampere)



A100 SXM

7 nm

Compute Capabilites **8.0**

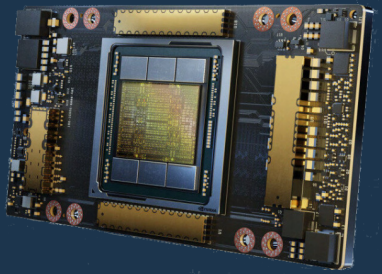
108 SMs **6912** Cores

Peak **19.5 TFlops** (without tensor cores)

40 - **80** GB DRAM

2020 - Ampere GPU

A100 (Ampere)



A100 SXM

7 nm

Compute Capabilites **8.0**

108 SMs **6912** Cores

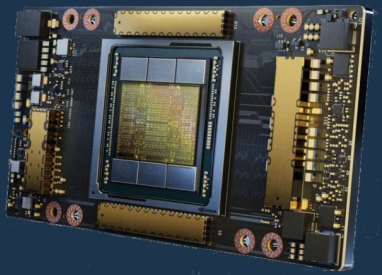
Peak **19.5 TFlops** (without tensor cores)

40 - **80** GB DRAM

1.3 TB/s Bandwidth (**HBM2**)

2020 - Ampere GPU

A100 (Ampere)



A100 SXM

7 nm

Compute Capabilites **8.0**

108 SMs **6912** Cores

Peak **19.5 TFlops** (without tensor cores)

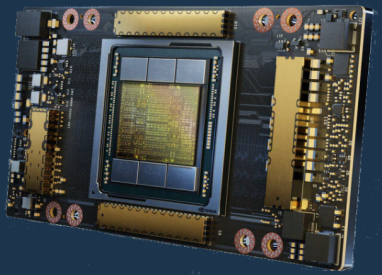
40 - **80** GB DRAM

1.3 TB/s Bandwidth (**HBM2**)

NVLink 600 GB/s

2020 - Ampere GPU

A100 (Ampere)



A100 SXM

7 nm

Compute Capabilites **8.0**

108 SMs **6912** Cores

Peak **19.5 TFlops** (without tensor cores)

40 - **80** GB DRAM

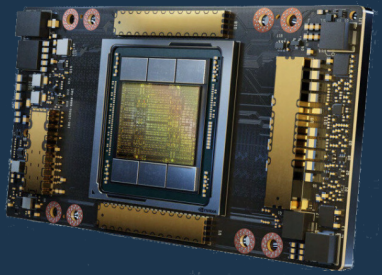
1.3 TB/s Bandwidth (**HBM2**)

NVLink 600 GB/s

MIG : **Multiple Instance GPU**

2020 - Ampere GPU

A100 (Ampere)



A100 SXM

7 nm

Compute Capabilites **8.0**

108 SMs **6912** Cores

Peak **19.5 TFlops** (without tensor cores)

40 - **80** GB DRAM

1.3 TB/s Bandwidth (**HBM2**)

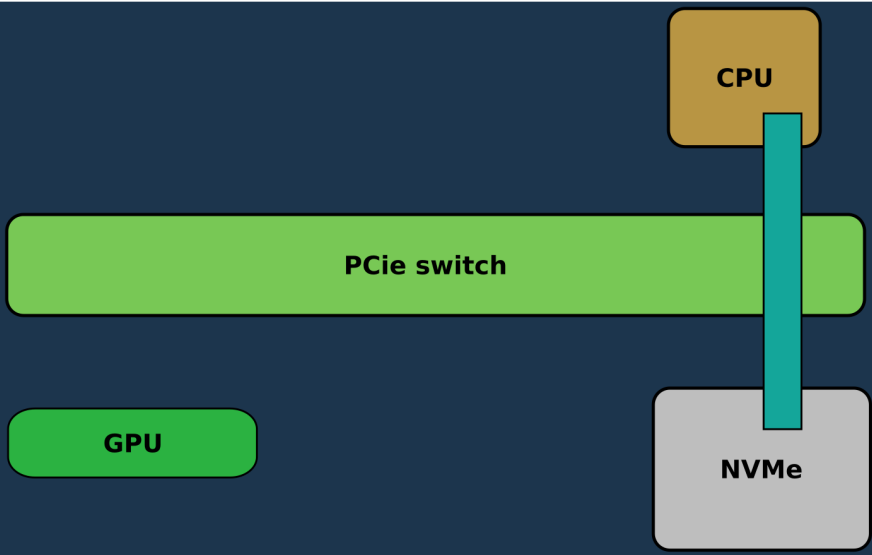
NVLink 600 GB/s

MIG : **Multiple Instance GPU**

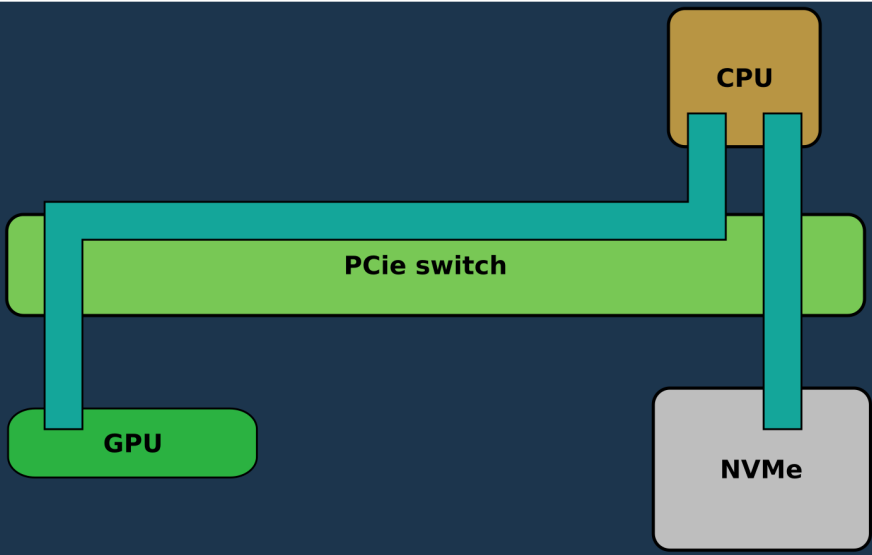
Too much for **PCIe** ?

2021 - Hardware evolution

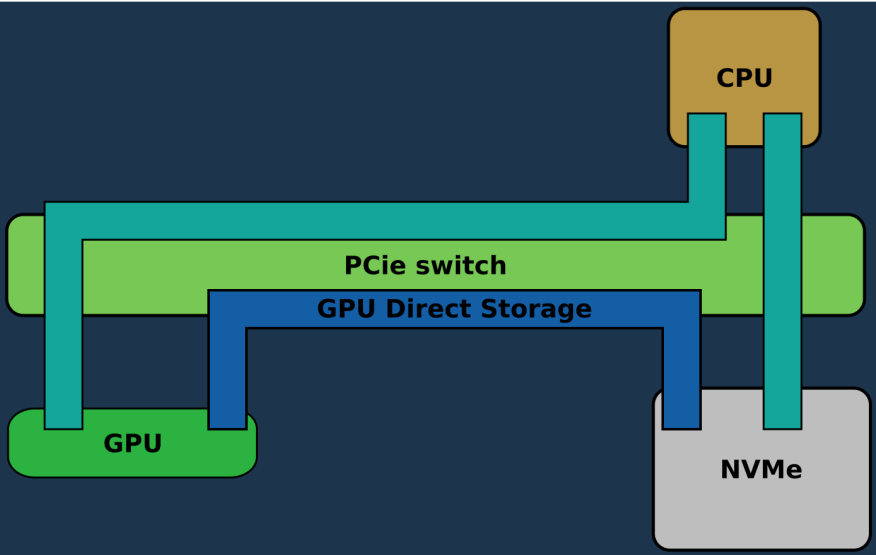
2021 - Hardware evolution



2021 - Hardware evolution

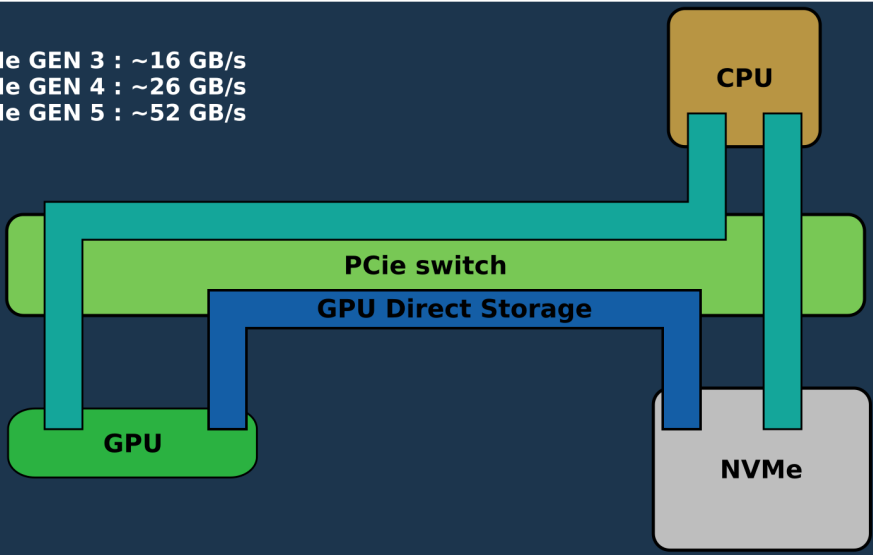


2021 - Hardware evolution



2021 - Hardware evolution

- PCIe GEN 3 : ~16 GB/s
- PCIe GEN 4 : ~26 GB/s
- PCIe GEN 5 : ~52 GB/s



NVCC, NVC++ evolution (NVidia)

NVCC, NVC++ evolution (NVidia)

Today (2021)

NVCC, NVC++ evolution (NVidia)

Today (2021)

nvc++

C++17 : **std::transform**
Fortran : **do concurent**

NVCC, NVC++ evolution (NVidia)

Today (2021)

nvc++

-stdpart
target GPU

C++17 : **std::transform**
Fortran : **do concurent**

NVCC, NVC++ evolution (NVidia)

Today (2021)

nvc++

-stdpart
target GPU

C++17 : **std::transform**
Fortran : **do concurent**

Compute Capabilities :
- Ampere : 8
- Turing : 7
- Volta : 7
- Pascal : 6

NVCC, NVC++ evolution (NVidia)

Today (2021)

nvc++

-stdpart
target GPU

C++17 : **std::transform**
Fortran : **do concurent**

Compute Capabilities :
- Ampere : 8
- Turing : 7
- Volta : 7
- Pascal : 6

nvcc

CUDA_ARCH

__host__
__device__
__global__

NVCC, NVC++ evolution (NVidia)

Today (2021)

Future (2022-2023)

nvc++

-stdpart
target GPU

C++17 : **std::transform**
Fortran : **do concurent**

Compute Capabilities :
- Ampere : 8
- Turing : 7
- Volta : 7
- Pascal : 6

nvcc

CUDA_ARCH

__host__
__device__
__global__

NVCC, NVC++ evolution (NVidia)

Today (2021)

nvc++

-stdpart
target GPU

C++17 : **std::transform**
Fortran : **do concurent**

Compute Capabilities :
- Ampere : 8
- Turing : 7
- Volta : 7
- Pascal : 6

nvcc

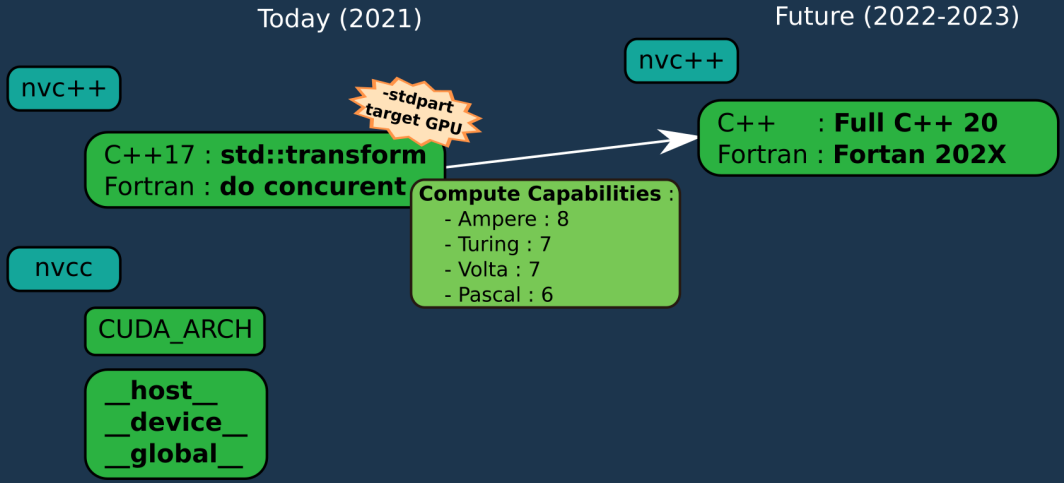
CUDA_ARCH

__host__
__device__
__global__

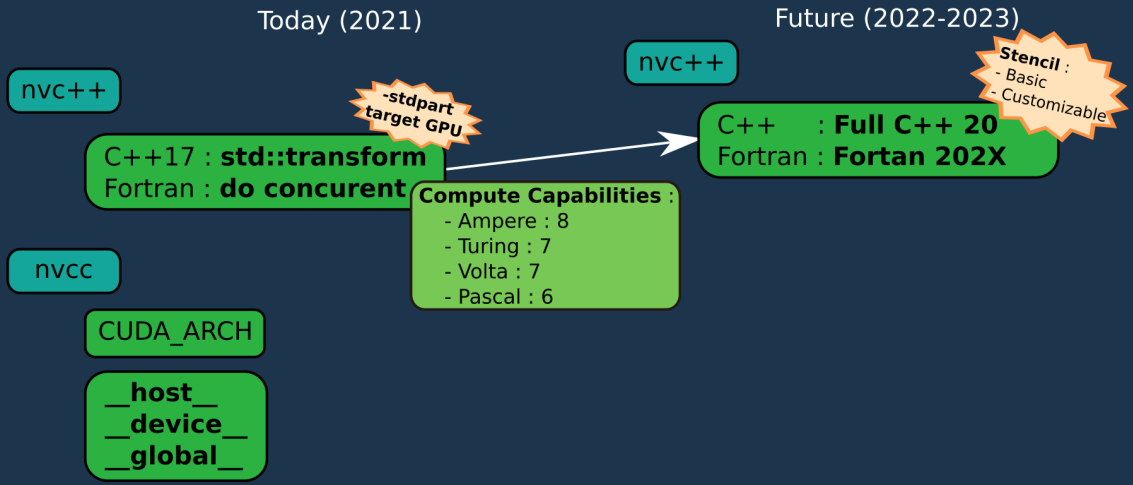
Future (2022-2023)

nvc++

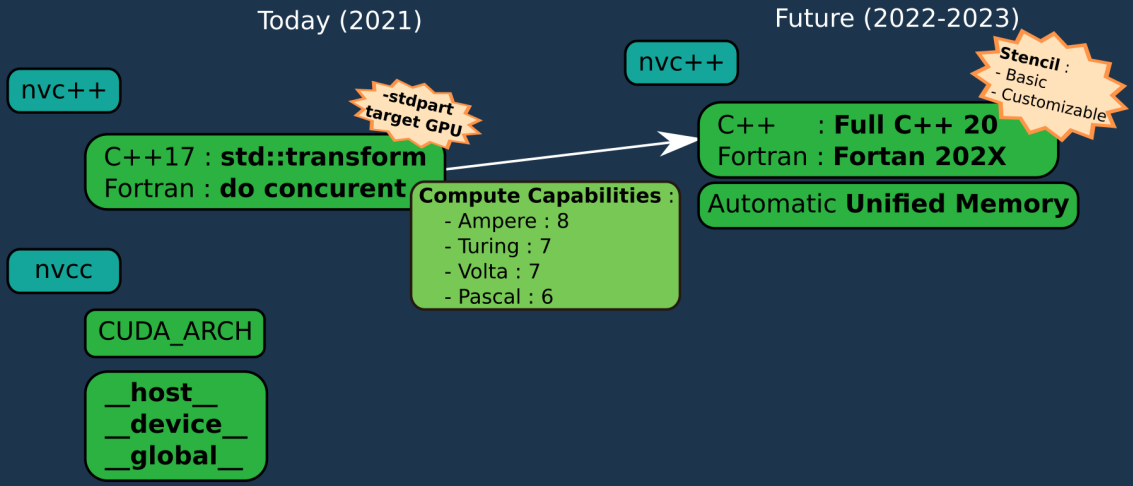
NVCC, NVC++ evolution (NVidia)



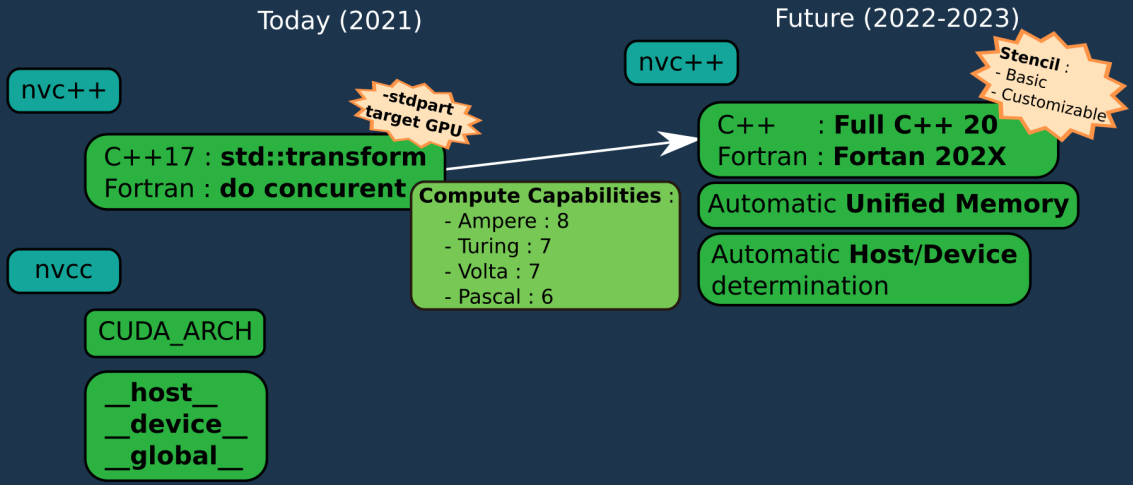
NVCC, NVC++ evolution (NVidia)



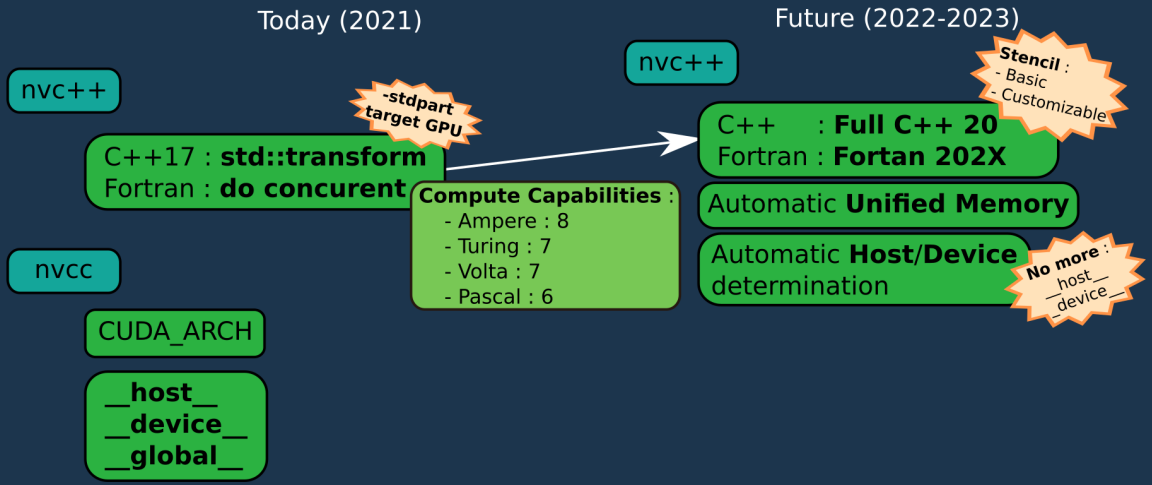
NVCC, NVC++ evolution (NVidia)



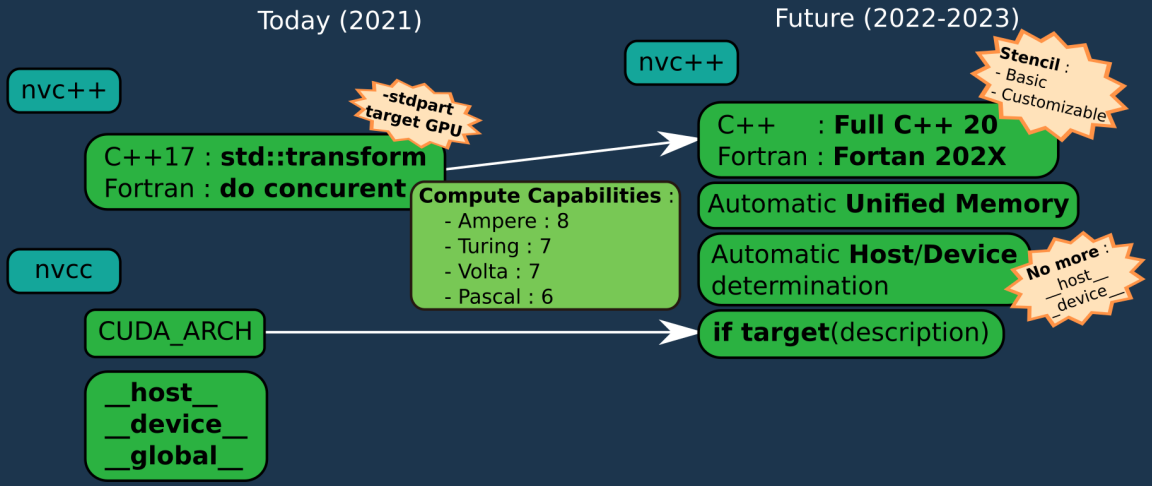
NVCC, NVC++ evolution (NVidia)



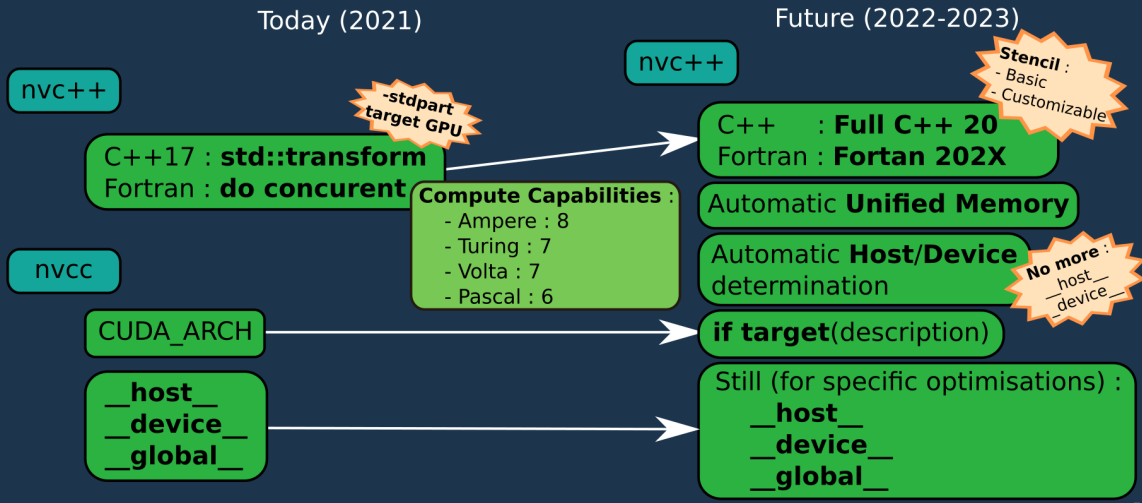
NVCC, NVC++ evolution (NVidia)



NVCC, NVC++ evolution (NVidia)



NVCC, NVC++ evolution (NVidia)



GPU and FORTRAN (NVFortran)

```
d = 2.5 * ceil(transpose(a)) + 3.0 * abs(transpose(b))
d = 2.5 * ceil(transpose(a)) + 3.0 * abs(b)
d = reshape(a,shape=[ni,nj,nk])
d = reshape(a,shape=[ni,nk,nj])
d = 2.5 * sqrt(reshape(a,shape=[ni,nk,nj],order=[1,3,2]))
d = alpha * conjg(reshape(a,shape=[ni,nk,nj],order=[1,3,2]))
d = reshape(a,shape=[ni,nk,nj],order=[1,3,2])
d = reshape(a,shape=[nk,ni,nj],order=[2,3,1])
d = reshape(a,shape=[ni*nj,nk])
d = reshape(a,shape=[nk,ni*nj],order=[2,1])
d = reshape(a,shape=[64,2,16,16,64],order=[5,2,3,4,1])
d = abs(reshape(a,shape=[64,2,16,16,64],order=[5,2,3,4,1]))
c = matmul(a,b)
c = matmul(transpose(a),b)
c = matmul(reshape(a,shape=[m,k],order=[2,1]),b)
c = matmul(a,transpose(b))
c = matmul(a,reshape(b,shape=[k,n],order=[2,1]))
```

```
c = matmul(transpose(a),transpose(b))
c = matmul(transpose(a),reshape(b,shape=[k,n],order=[2,1]))
d = spread(a,dim=3,ncopies=nk)
d = spread(a,dim=1,ncopies=ni)
d = spread(a,dim=2,ncopies=nx)
d = alpha * abs(spread(a,dim=2,ncopies=nx))
d = alpha * spread(a,dim=2,ncopies=nx)
d = abs(spread(a,dim=2,ncopies=nx))
d = transpose(a)
d = alpha * transpose(a)
d = alpha * ceil(transpose(a))
d = alpha * conjg(transpose(a))
c = c + matmul(a,b)
c = c - matmul(a,b)
c = c + alpha * matmul(a,b)
d = alpha * matmul(a,b) + c
d = alpha * matmul(a,b) + beta * c
```

Paradigms evolution

Current default situation

New default situation

Paradigms evolution

Current default situation

Scalar

New default situation

Paradigms evolution

Current default situation

Scalar

Single-threaded

New default situation

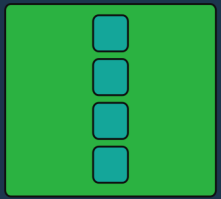
Paradigms evolution

Current default situation

Scalar

Single-threaded

Execution **in order**



New default situation

Paradigms evolution

Current default situation

Scalar

Single-threaded

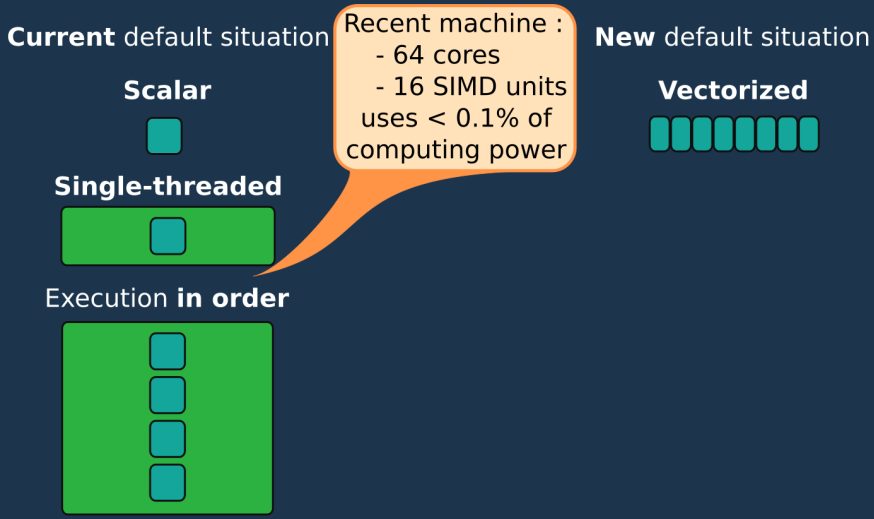
Execution **in order**

Recent machine :
- 64 cores
- 16 SIMD units

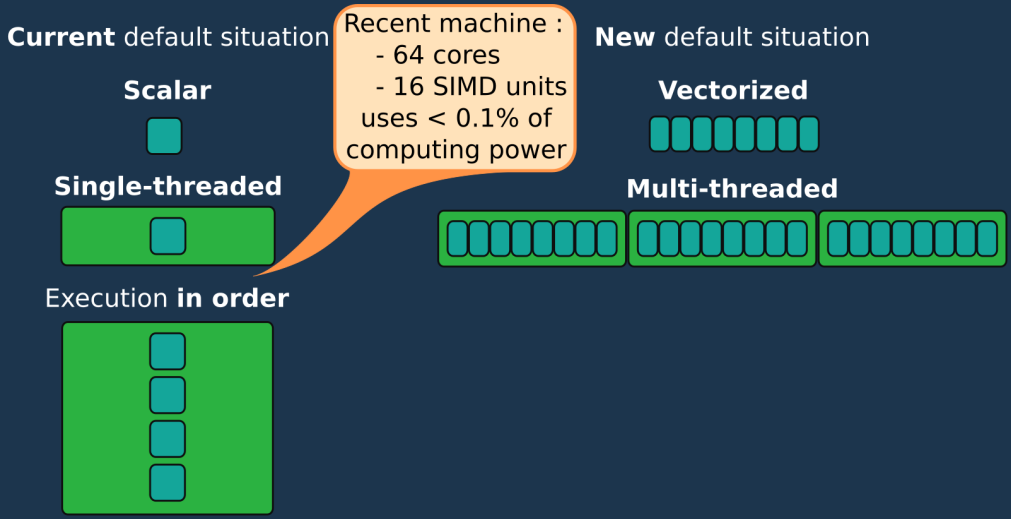
uses < 0.1% of computing power

New default situation

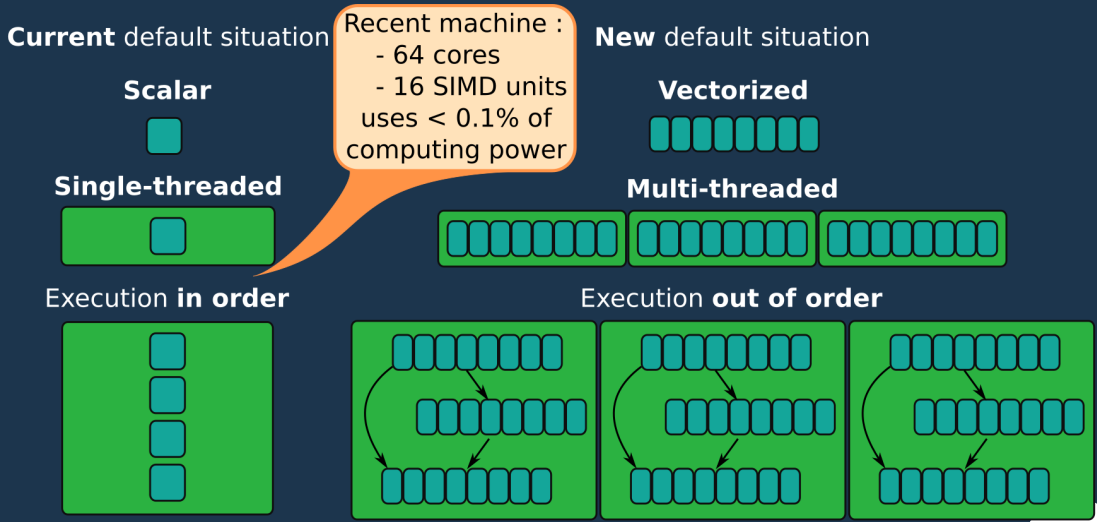
Paradigms evolution



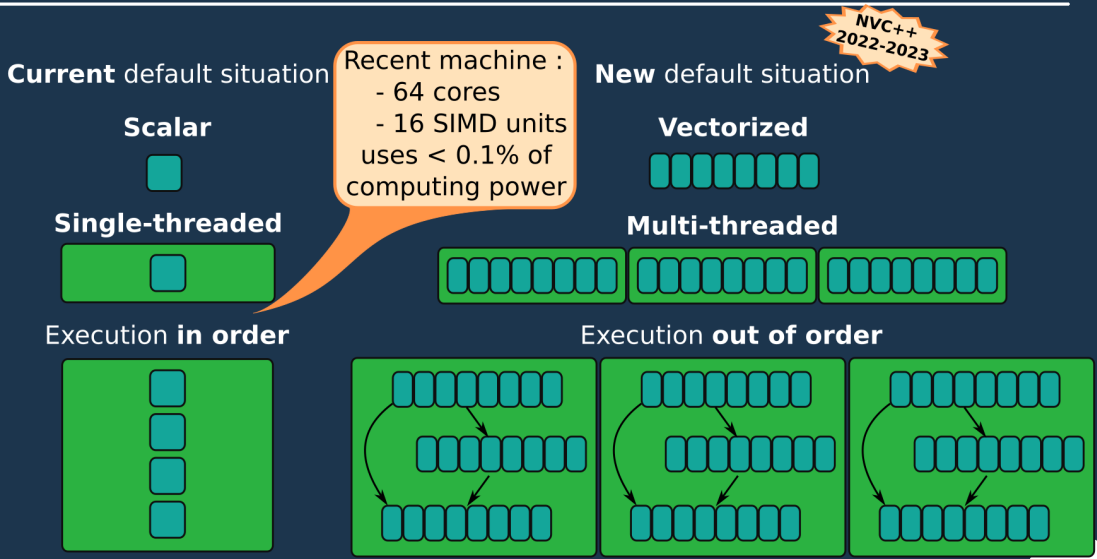
Paradigms evolution



Paradigms evolution



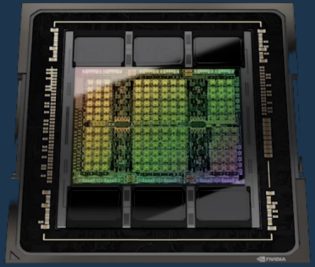
Paradigms evolution



2022 - Hopper GPU

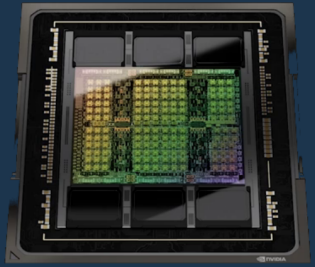
2022 - Hopper GPU

H100 (Hopper)



2022 - Hopper GPU

H100 (Hopper)

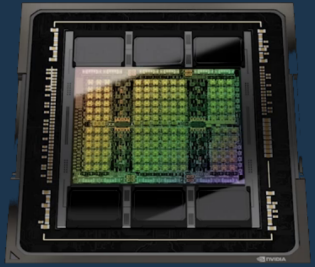


H100 SXM5 and PCie

2022 - Hopper GPU

4 nm (A100 7 nm)

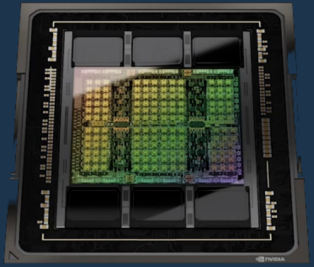
H100 (Hopper)



H100 SXM5 and PCie

2022 - Hopper GPU

H100 (Hopper)



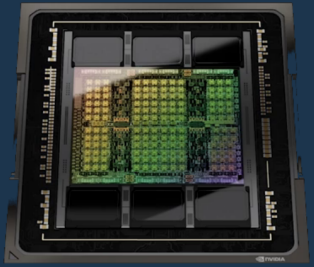
H100 SXM5 and PCie

4 nm (A100 7 nm)

Compute Capabilites 9.0

2022 - Hopper GPU

H100 (Hopper)



H100 SXM5 and PCie

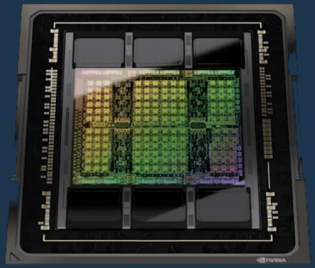
4 nm (A100 7 nm)

Compute Capabilites 9.0

132 SMs (tensor core Gen4) 8448 Cores

2022 - Hopper GPU

H100 (Hopper)



H100 SXM5 and PCie

4 nm (A100 7 nm)

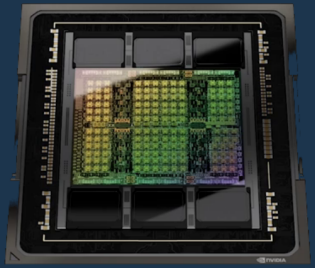
Compute Capabilites 9.0

132 SMs (tensor core Gen4) 8448 Cores

80 GB DRAM

2022 - Hopper GPU

H100 (Hopper)



H100 SXM5 and PCie

4 nm (A100 7 nm)

Compute Capabilites 9.0

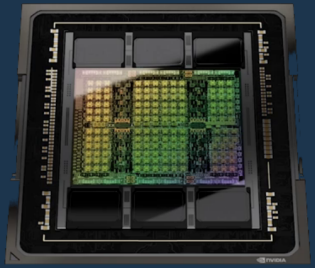
132 SMs (tensor core Gen4) 8448 Cores

80 GB DRAM

3 TB/s Bandwidth (HBM3)

2022 - Hopper GPU

H100 (Hopper)



H100 SXM5 and PCie

4 nm (A100 7 nm)

Compute Capabilites 9.0

132 SMs (tensor core Gen4) 8448 Cores

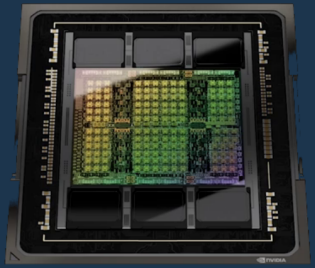
80 GB DRAM

3 TB/s Bandwidth (HBM3)

NVLink Gen 4 (900 GB/s)

2022 - Hopper GPU

H100 (Hopper)

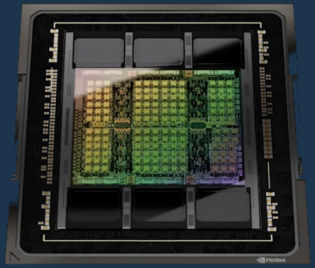


H100 SXM5 and PCie

- 4 nm (A100 7 nm)
- Compute Capabilites 9.0
- 132 SMs (tensor core Gen4) 8448 Cores
- 80 GB DRAM
- 3 TB/s Bandwidth (HBM3)
- NVLink Gen 4 (900 GB/s)
- MIG Gen 2 with confidential computing
  (all instances have Image/Video decoding)

2022 - Hopper GPU

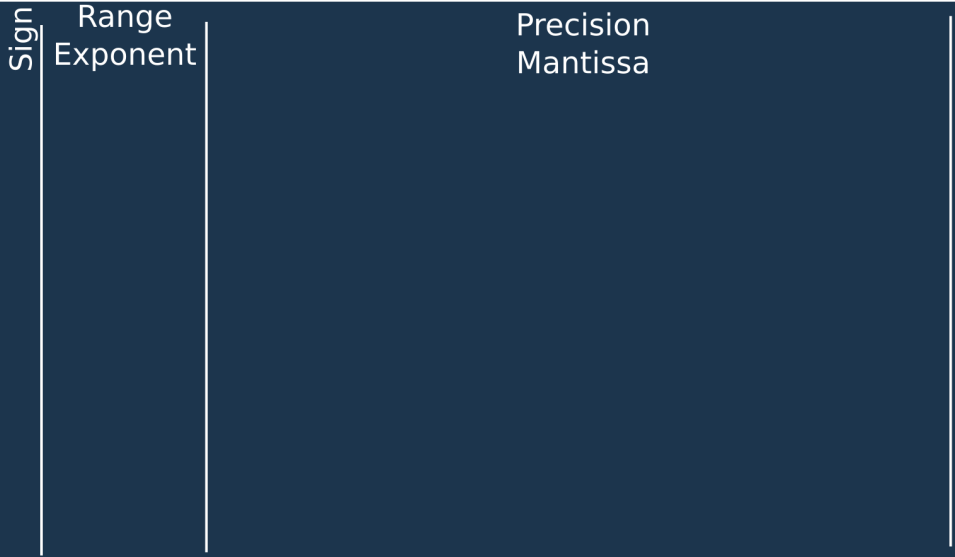
H100 (Hopper)



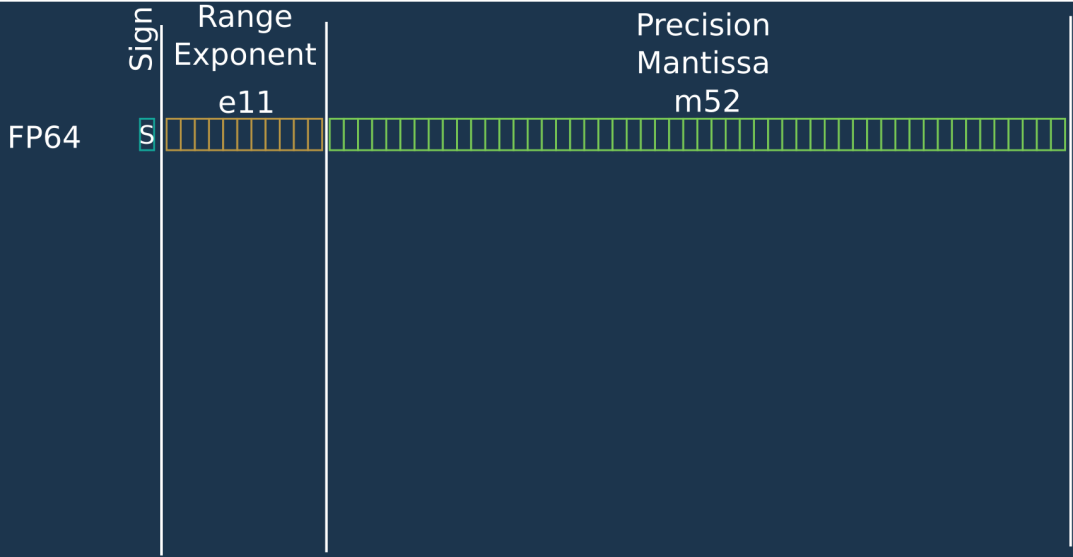
H100 SXM5 and PCie

- 4 nm (A100 7 nm)
- Compute Capabilites 9.0
- 132 SMs (tensor core Gen4) 8448 Cores
- 80 GB DRAM
- 3 TB/s Bandwidth (HBM3)
- NVLink Gen 4 (900 GB/s)
- MIG Gen 2 with confidential computing
  (all instances have Image/Video decoding)
- A lot a computing precisions

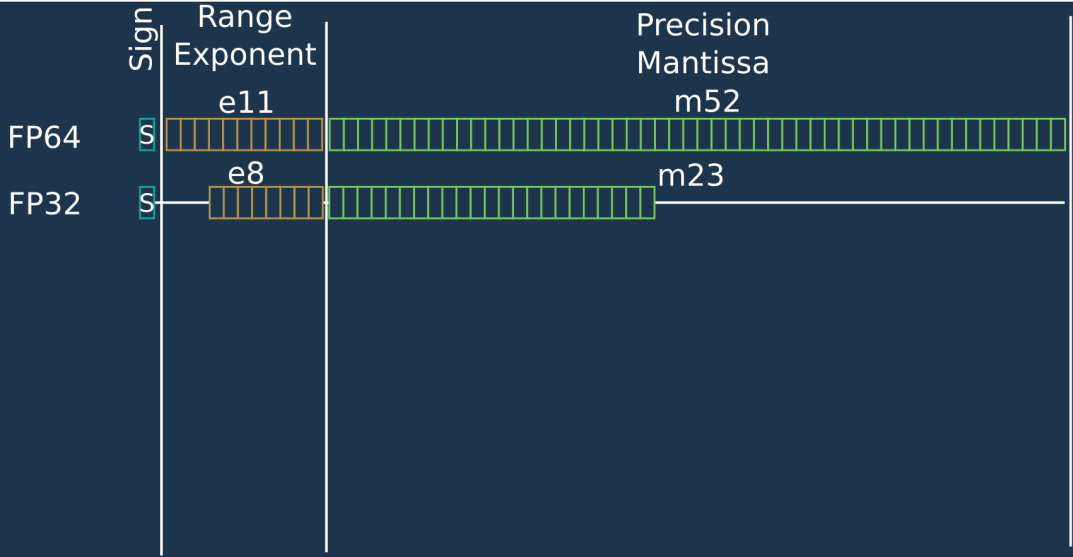
Computing Precision



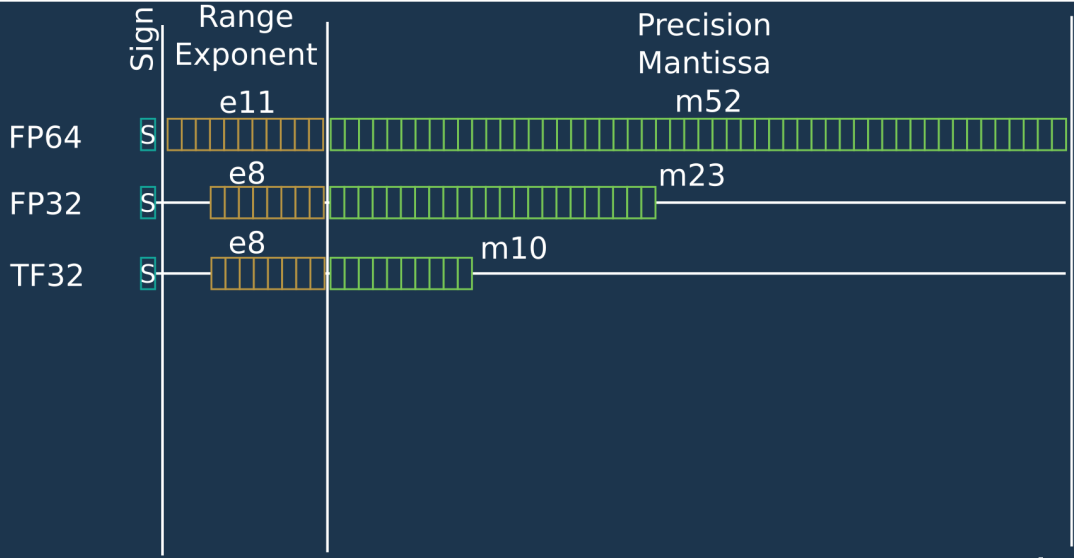
Computing Precision



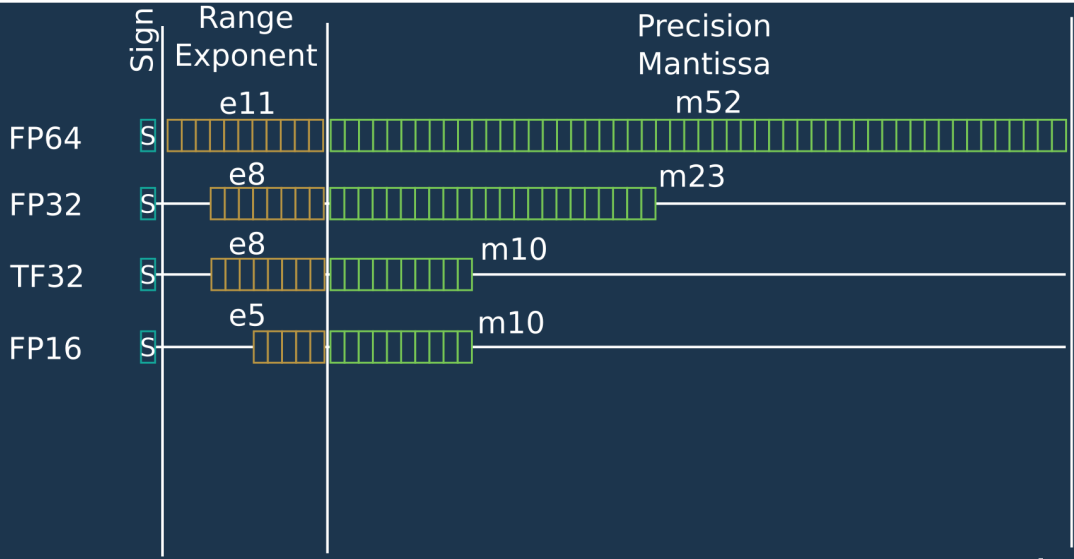
Computing Precision



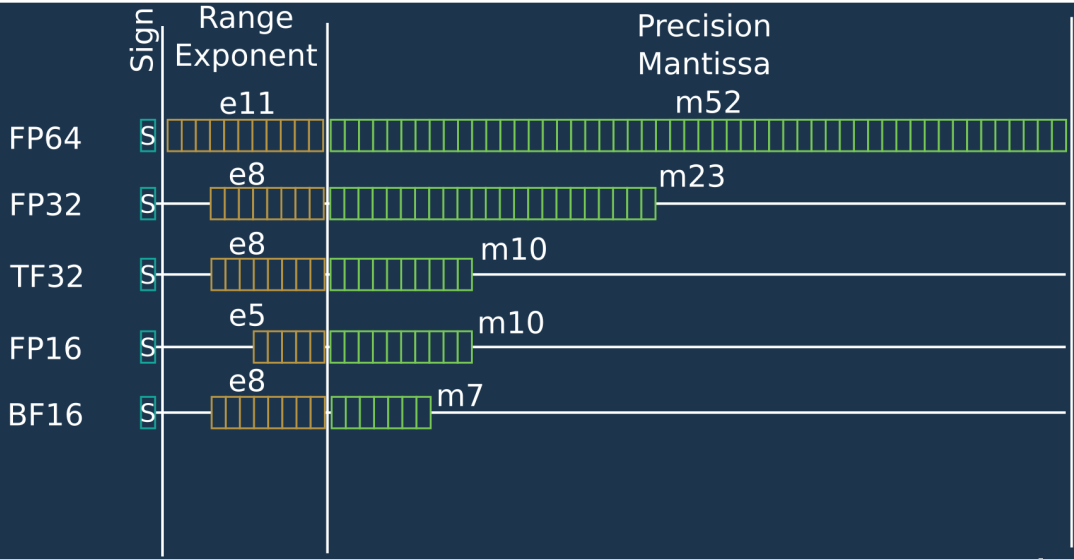
Computing Precision



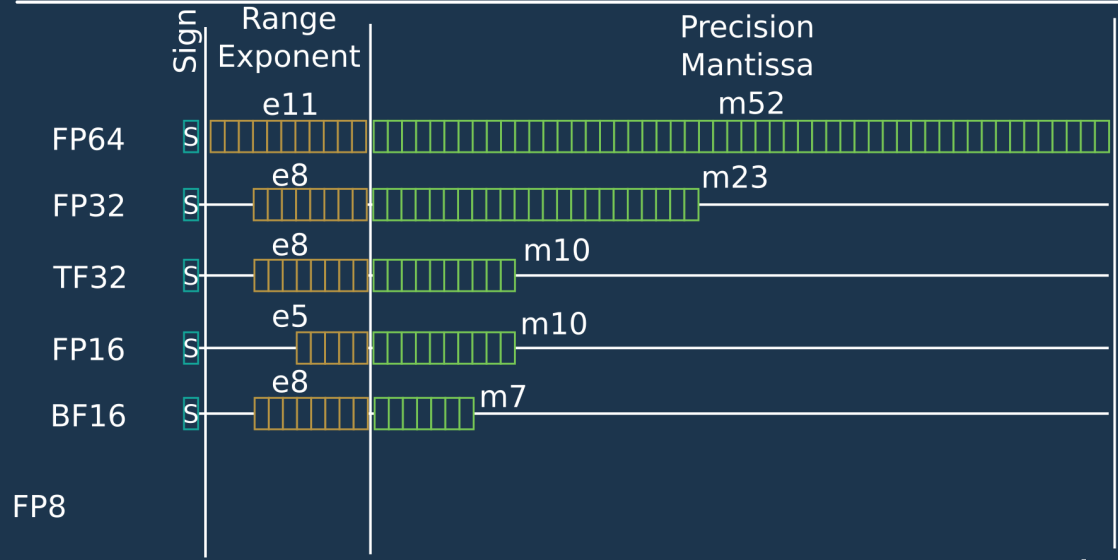
Computing Precision



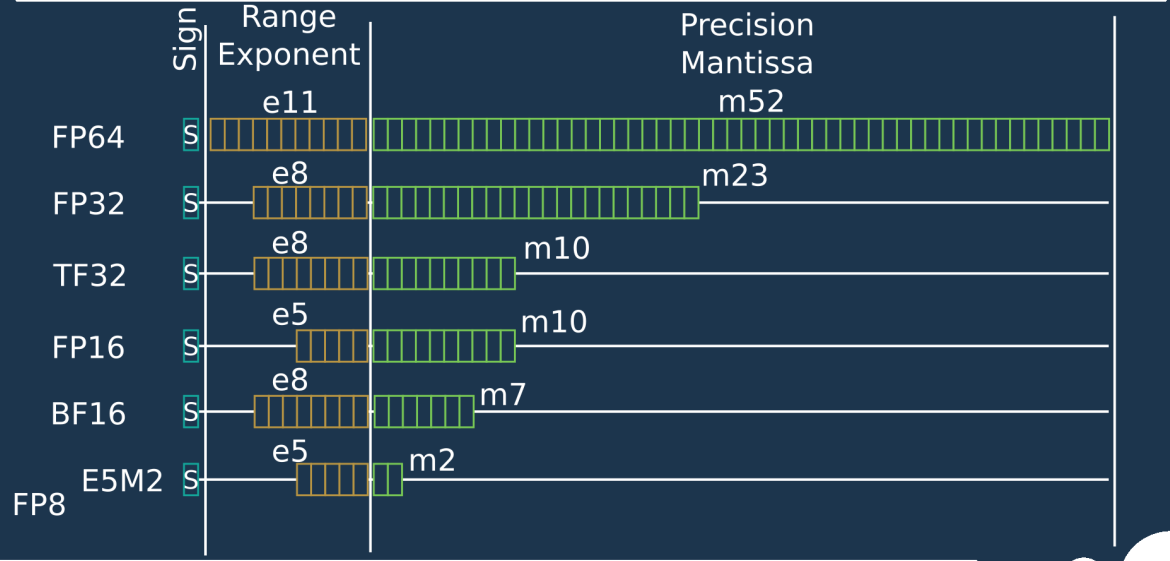
Computing Precision



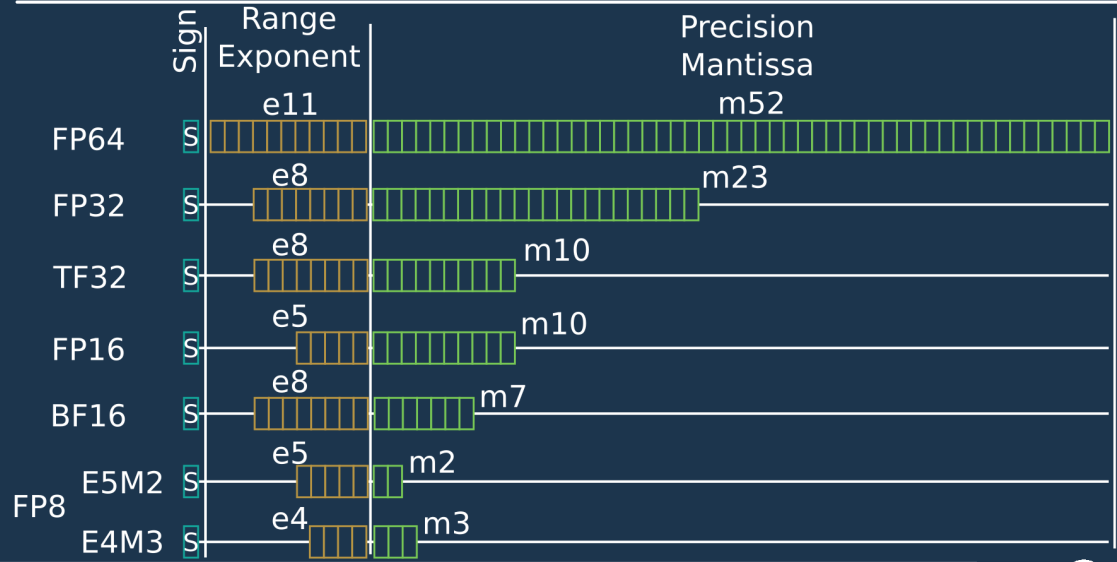
Computing Precision



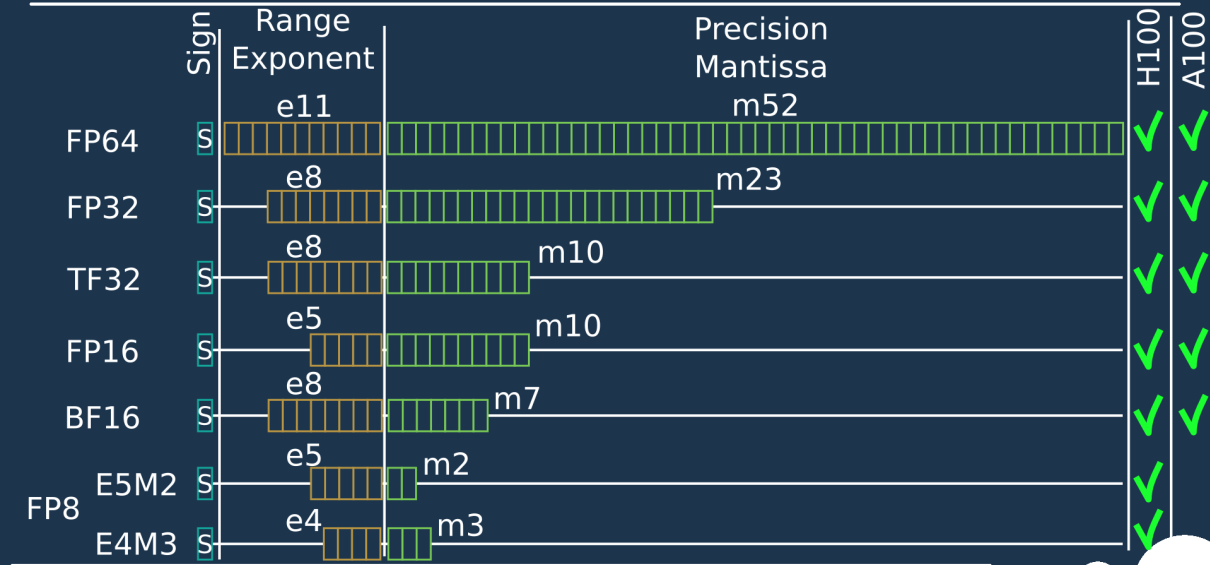
Computing Precision



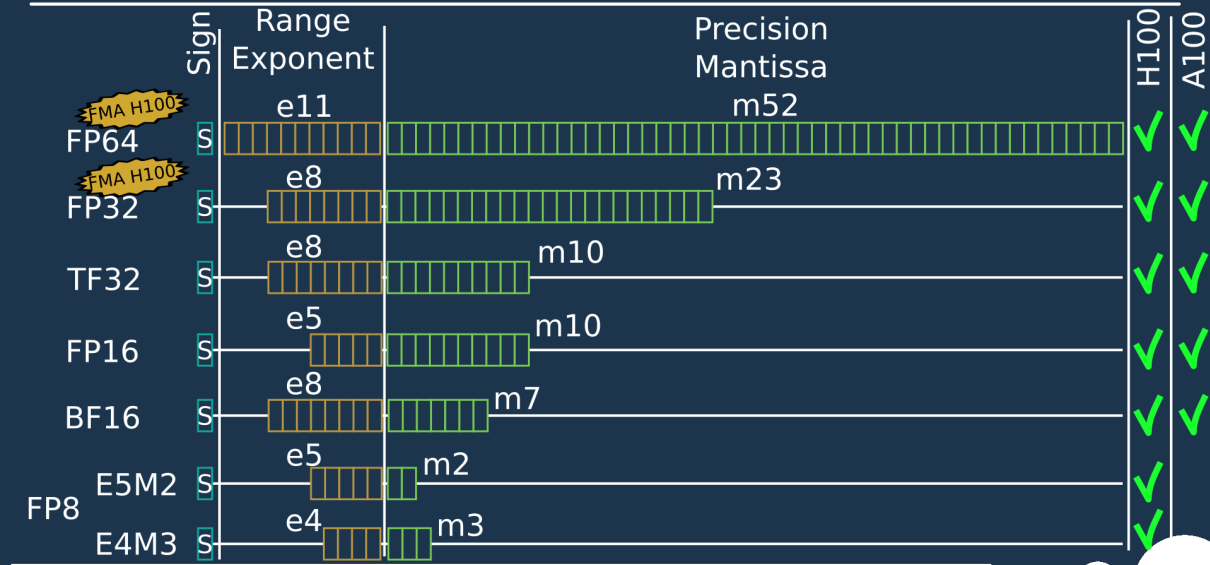
Computing Precision



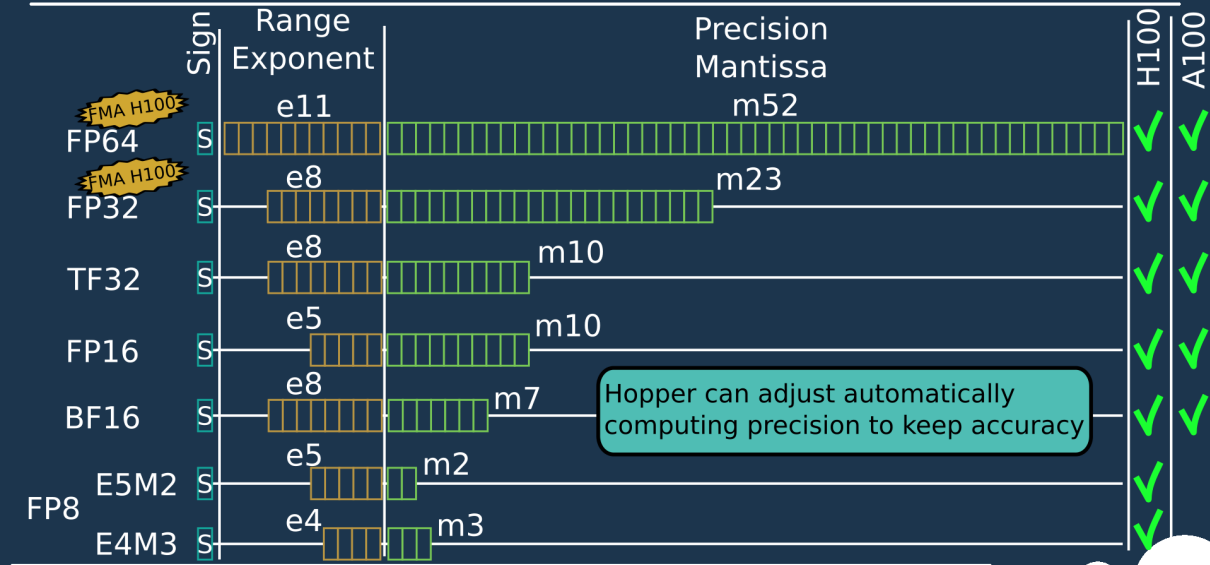
Computing Precision



Computing Precision



Computing Precision

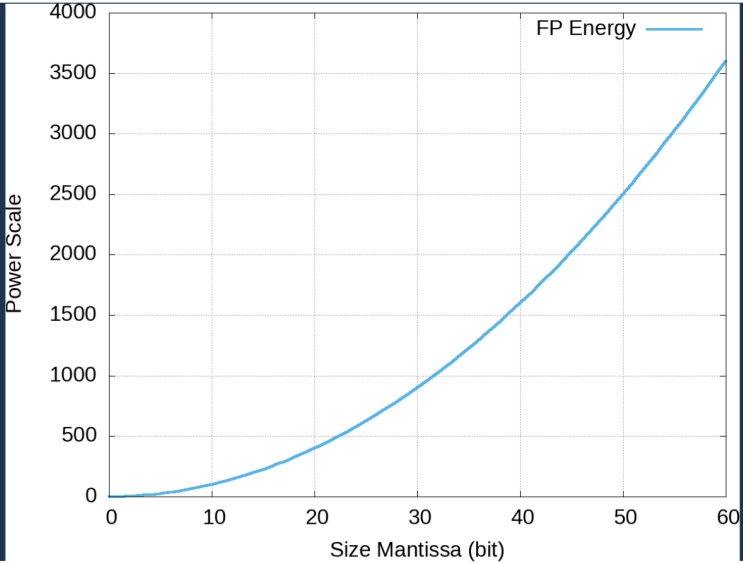


Why all these computing precisions ?

Why all these computing precisions ?

Lower precision are **fast** to compute

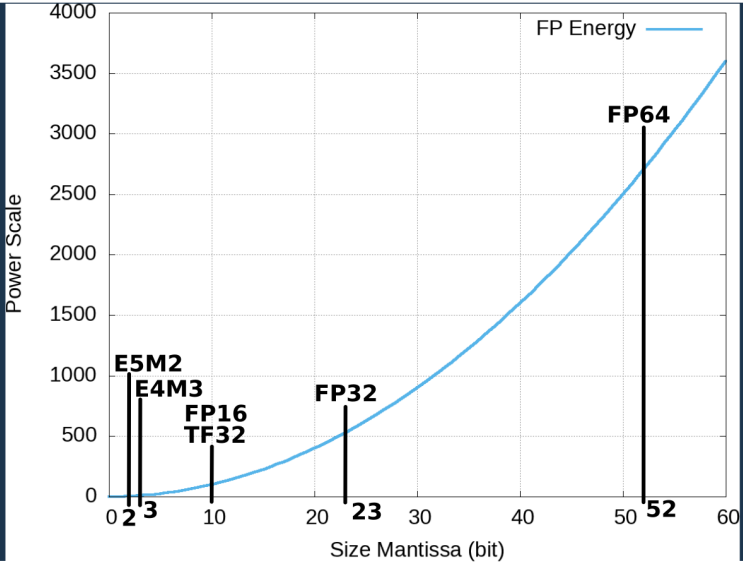
Why all these computing precisions ?



Lower precision are **fast** to compute

Consuption is mostly **proportional** to the **square** of **mantissa** size

Why all these computing precisions ?



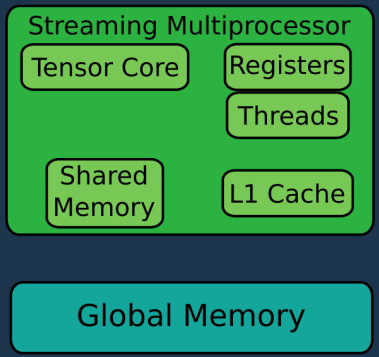
Lower precision are **fast** to compute

Consuption is mostly **proportional** to the **square** of **mantissa** size

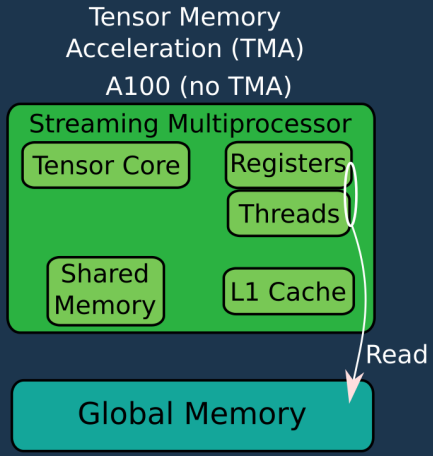
Thread acceleration

Thread acceleration

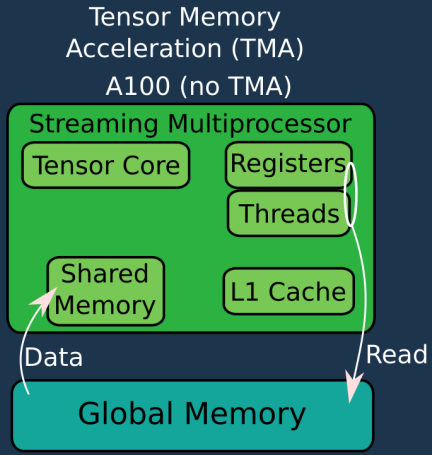
Tensor Memory Acceleration (TMA)



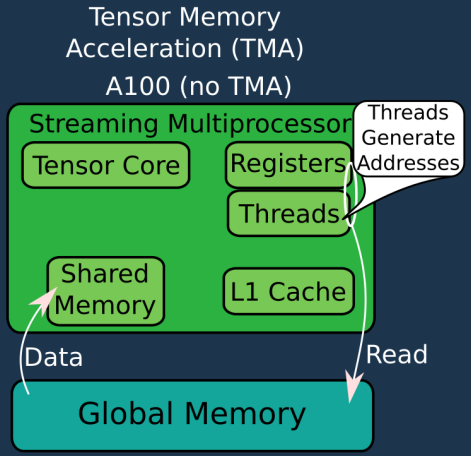
Thread acceleration



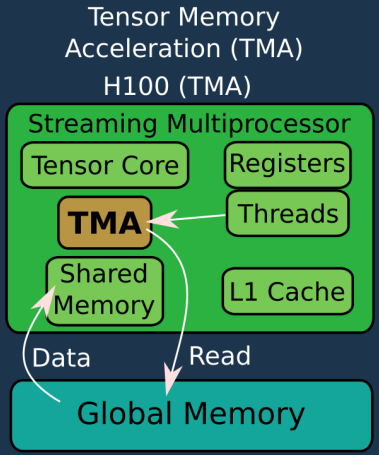
Thread acceleration



Thread acceleration



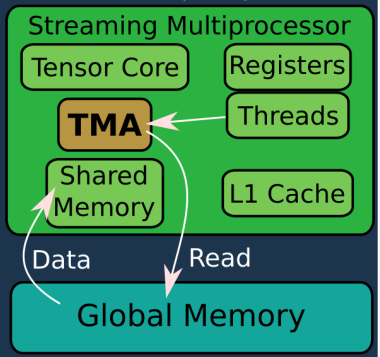
Thread acceleration



Thread acceleration

Tensor Memory Acceleration (TMA)

H100 (TMA)

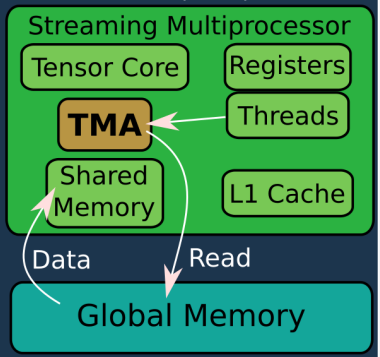


Distributed shared memory

Thread acceleration

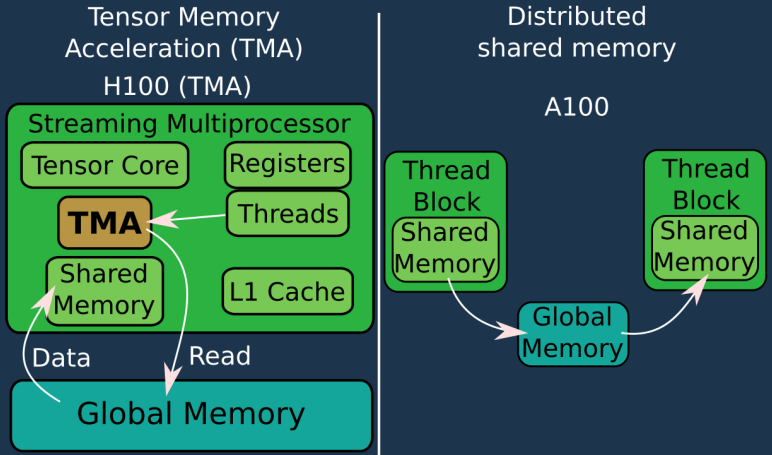
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

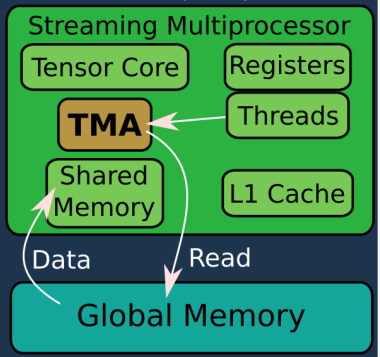
Thread acceleration



Thread acceleration

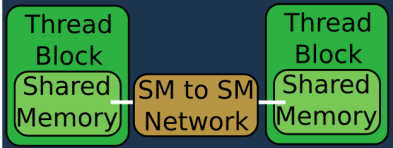
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

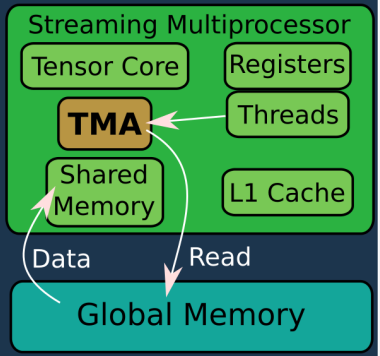
H100



Thread acceleration

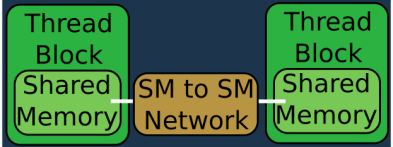
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

H100

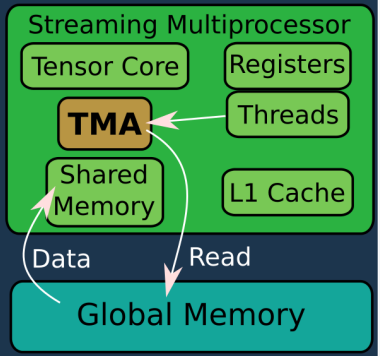


Asynchronous Transaction Barrier

Thread acceleration

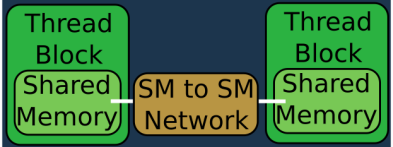
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

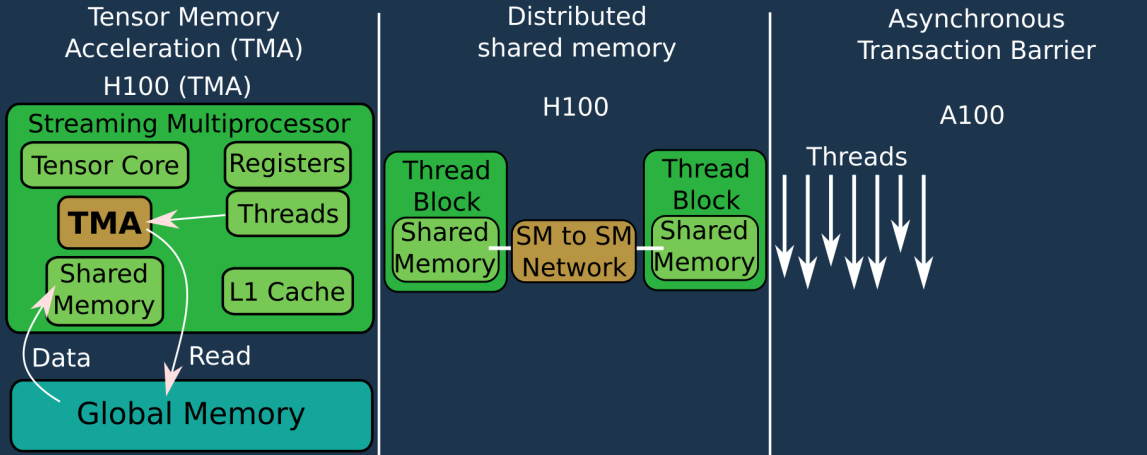
H100



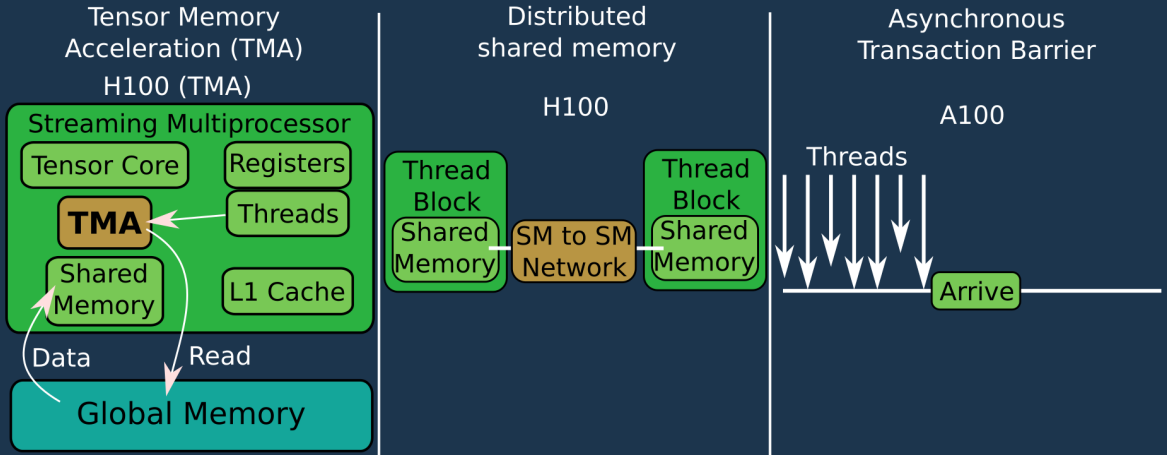
Asynchronous Transaction Barrier

A100

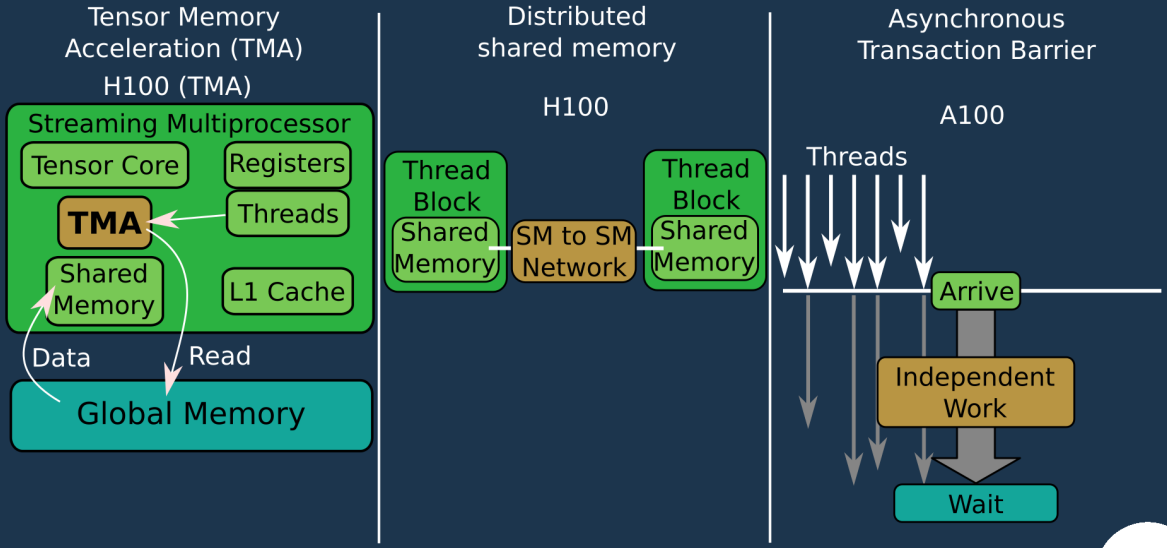
Thread acceleration



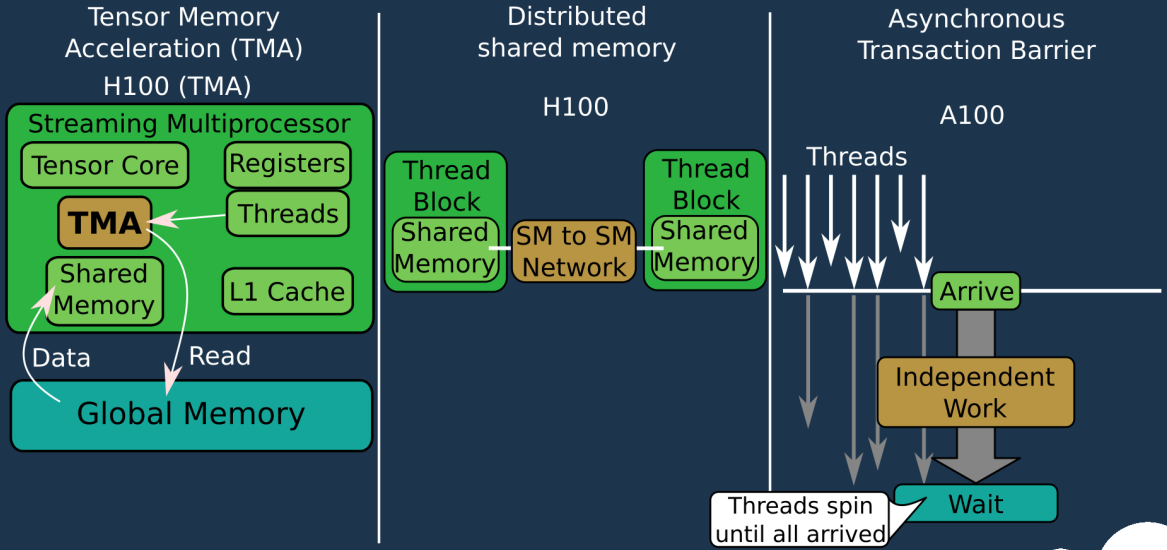
Thread acceleration



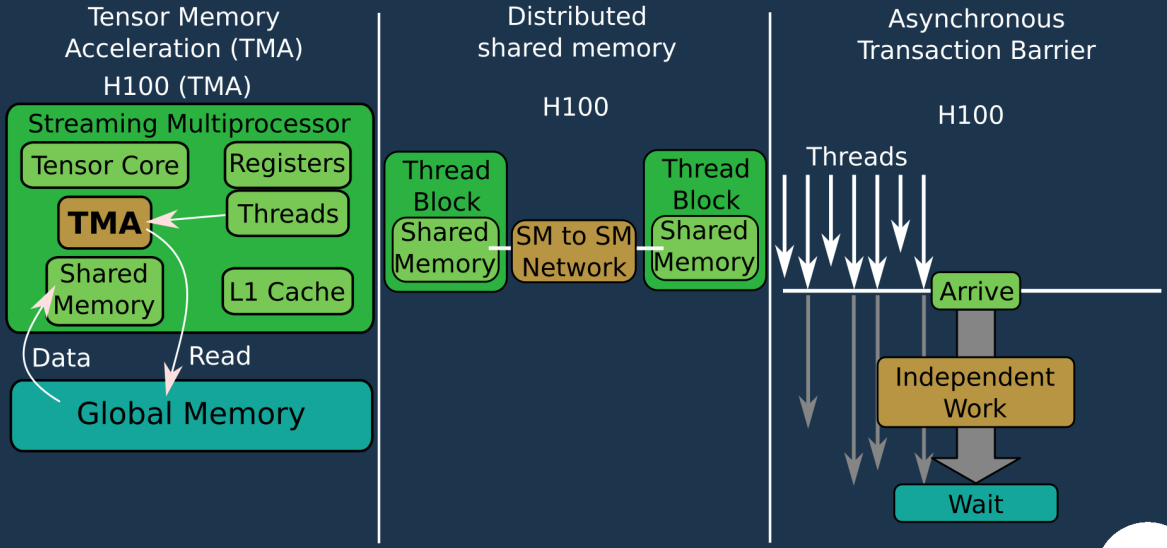
Thread acceleration



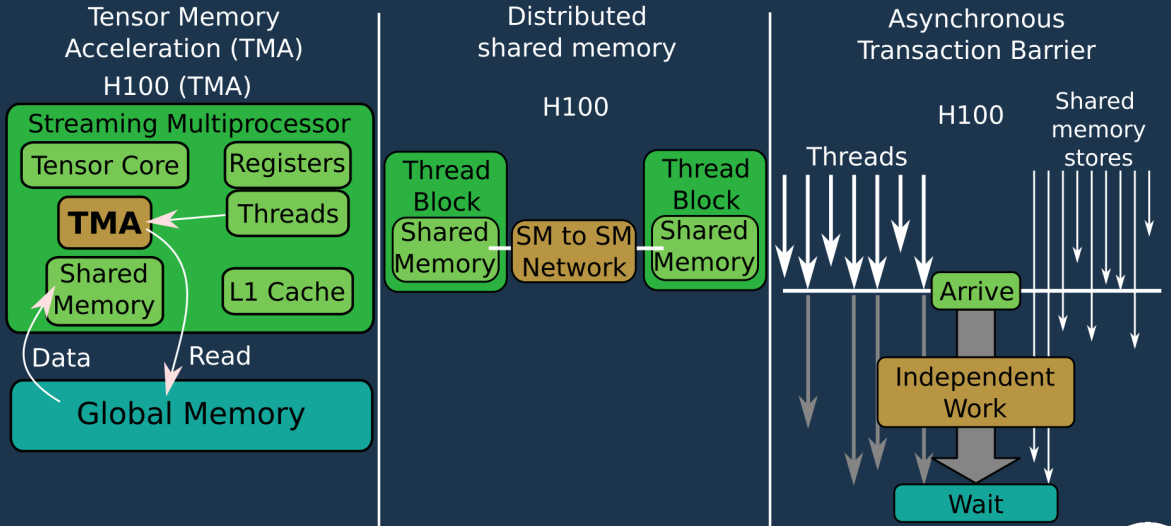
Thread acceleration



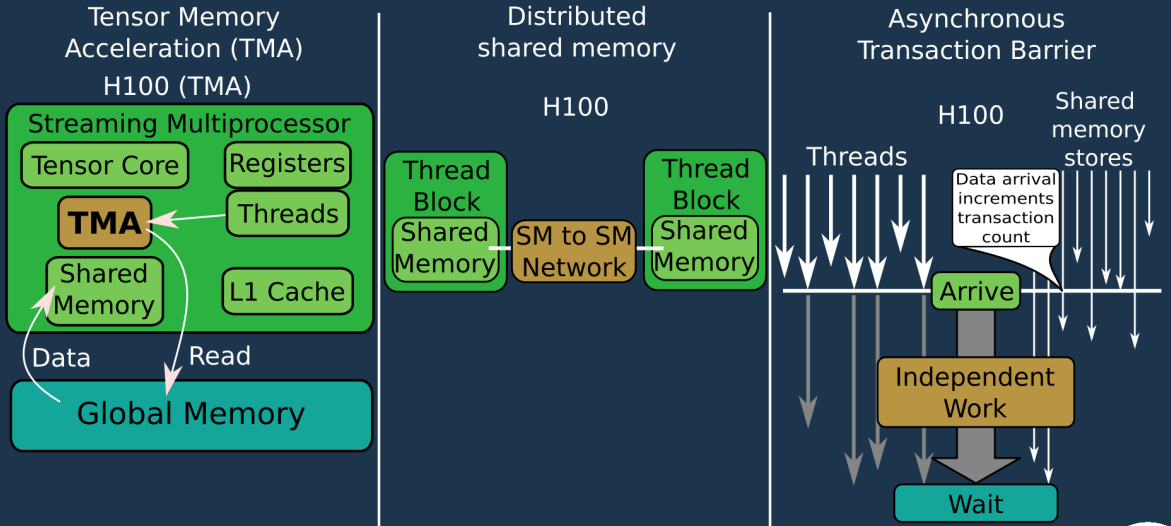
Thread acceleration



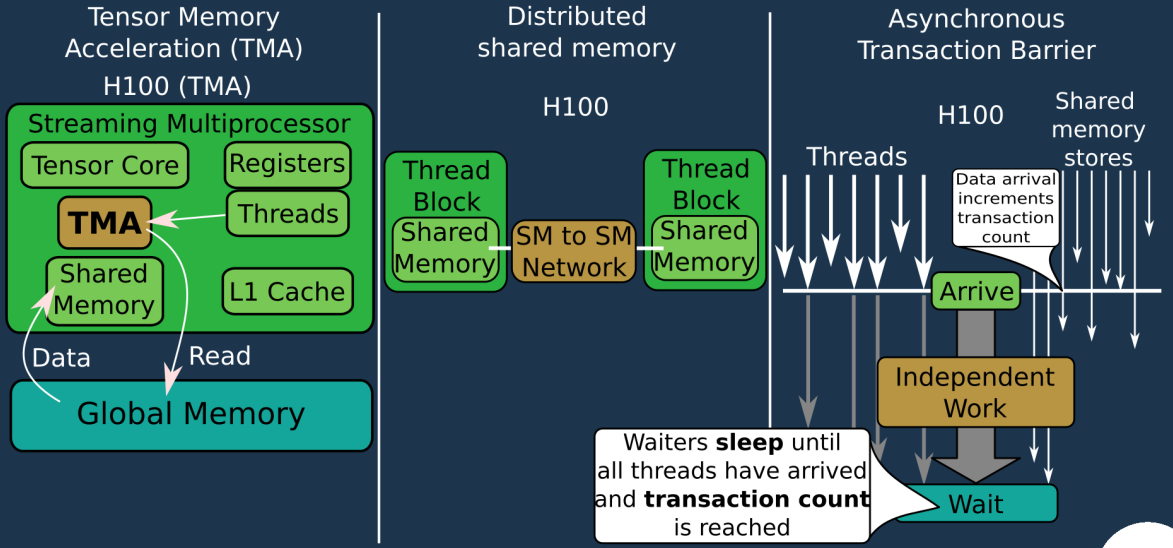
Thread acceleration



Thread acceleration



Thread acceleration



New intrinsic functions

New intrinsic functions

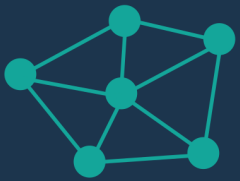
Dynamic Programming Acceleration (DPX) : recursive algorithms

New intrinsic functions

Dynamic Programming Acceleration (DPX) : recursive algorithms

Floyd-Warshall Algorithm

Route optimization



New intrinsic functions

Dynamic Programming Acceleration (DPX) : recursive algorithms

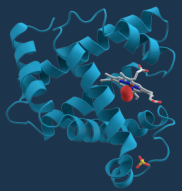
Floyd-Warshall Algorithm

Route optimization

Smith-Waterman and **Needleman-Wunsch** Algorithm

Pattern Matching

- DNA sequence alignment
- Protein classification
- Protein folding



New intrinsic functions

Dynamic Programming Acceleration (DPX) : recursive algorithms

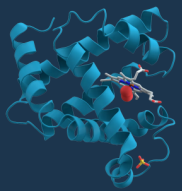
Floyd-Warshall Algorithm

Route optimization

Smith-Waterman and **Needleman-Wunsch** Algorithm

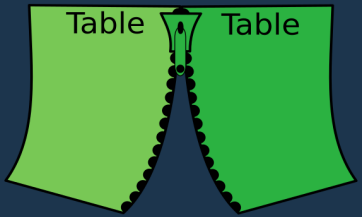
Pattern Matching

- DNA sequence alignment
- Protein classification
- Protein folding



SQL Query optimisation

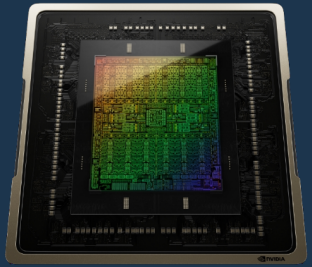
Join in optimal order



Hardware evolution

Hardware evolution

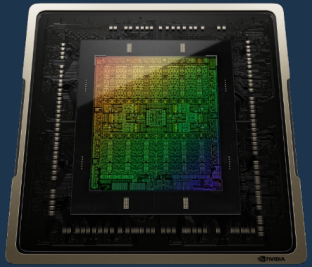
GeForce 4090
(Ada Lovelace)



Hardware evolution

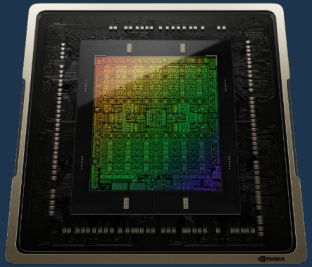
4 nm (A100 7 nm)

GeForce 4090
(Ada Lovelace)



Hardware evolution

GeForce 4090
(Ada Lovelace)

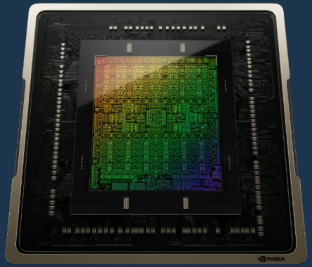


4 nm (A100 7 nm)

128 SMs (tensor core Gen4) 16384 Cuda Cores

Hardware evolution

GeForce 4090
(Ada Lovelace)



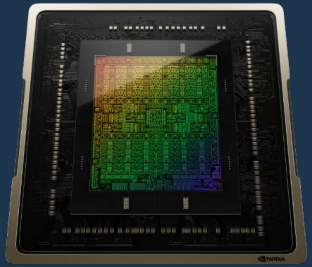
4 nm (A100 7 nm)

128 SMs (tensor core Gen4) 16384 Cuda Cores

24 GB GDDR6

Hardware evolution

GeForce 4090
(Ada Lovelace)



4 nm (A100 7 nm)

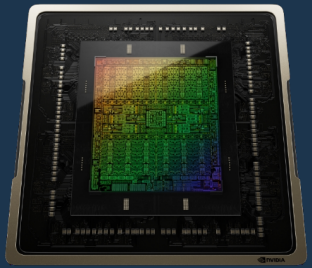
128 SMs (tensor core Gen4) 16384 Cuda Cores

24 GB GDDR6

1 TB/s Bandwidth

Hardware evolution

GeForce 4090
(Ada Lovelace)



4 nm (A100 7 nm)

128 SMs (tensor core Gen4) 16384 Cuda Cores

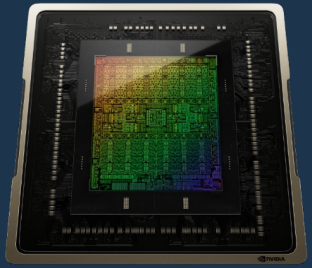
24 GB GDDR6

1 TB/s Bandwidth

Peak 82 TFlops (FP32 without tensor cores)

Hardware evolution

GeForce 4090
(Ada Lovelace)



4 nm (A100 7 nm)

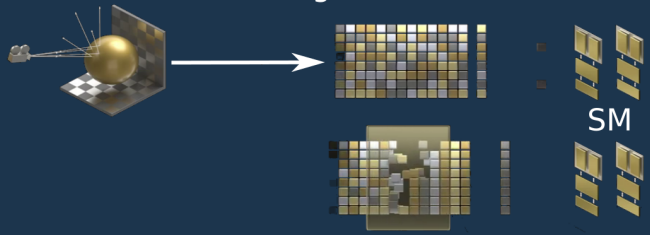
128 SMs (tensor core Gen4) 16384 Cuda Cores

24 GB GDDR6

1 TB/s Bandwidth

Peak 82 TFlops (FP32 without tensor cores)

Shader Reordering



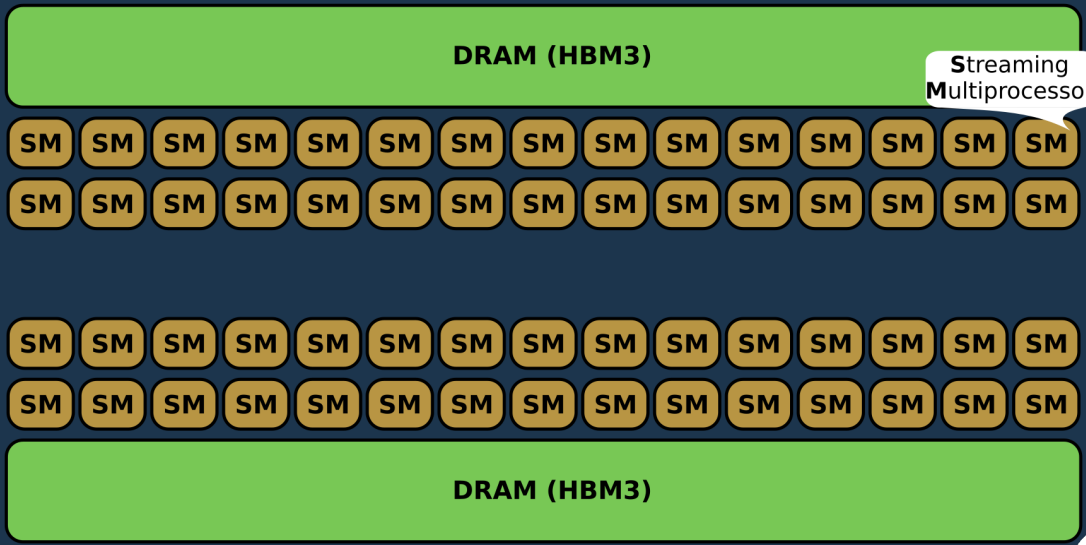
GPU Architecture

GPU Architecture

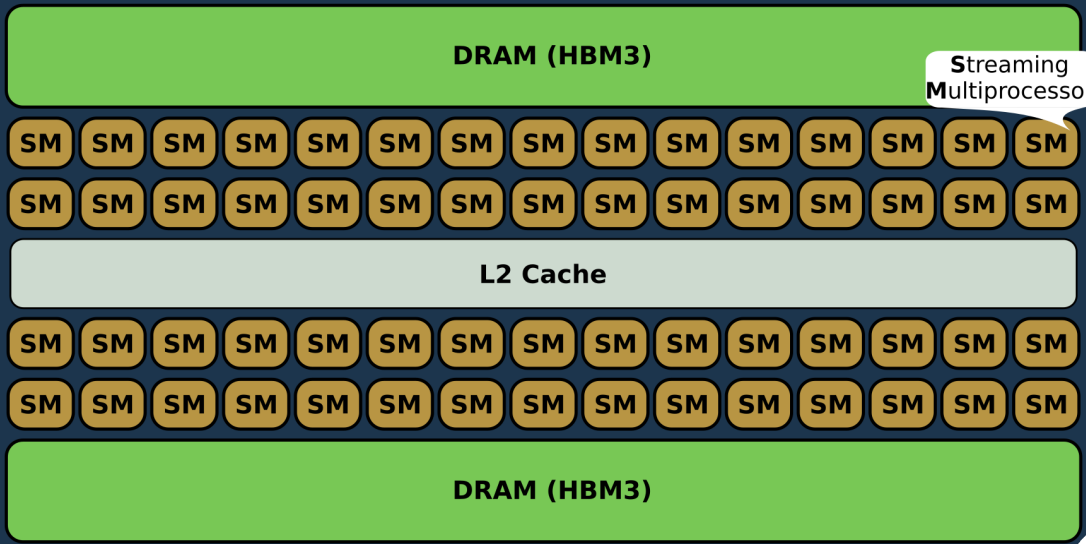
DRAM (HBM3)

DRAM (HBM3)

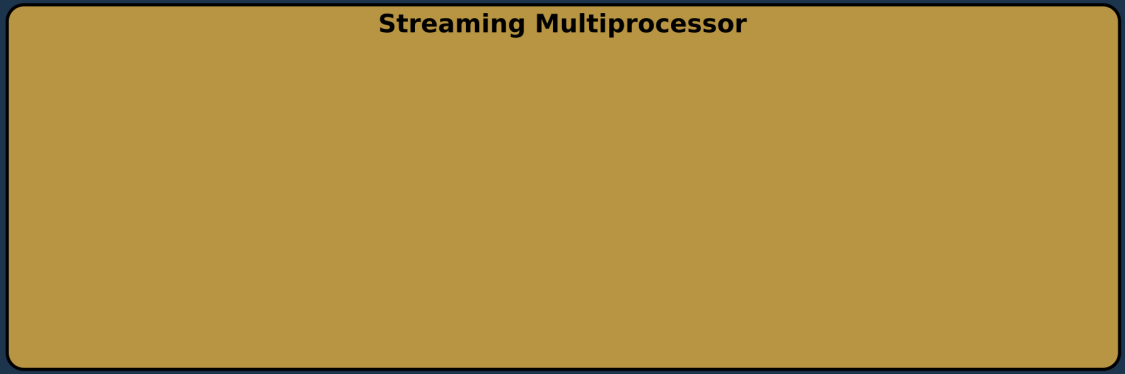
GPU Architecture



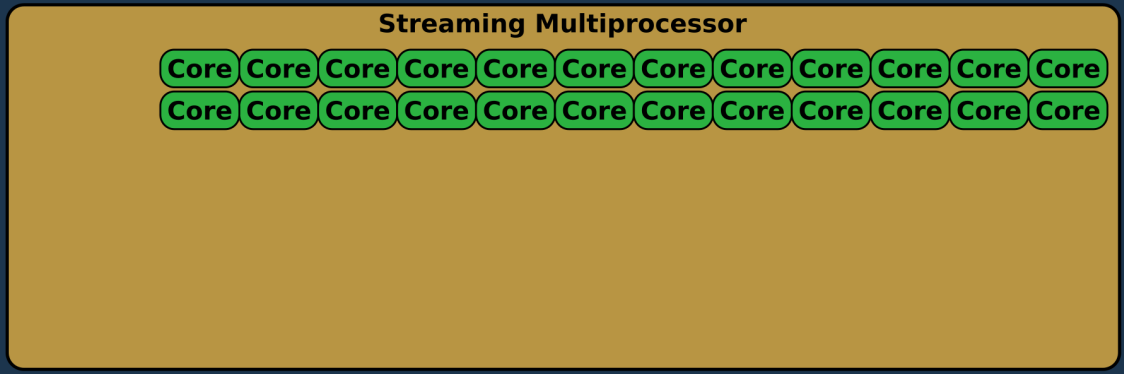
GPU Architecture



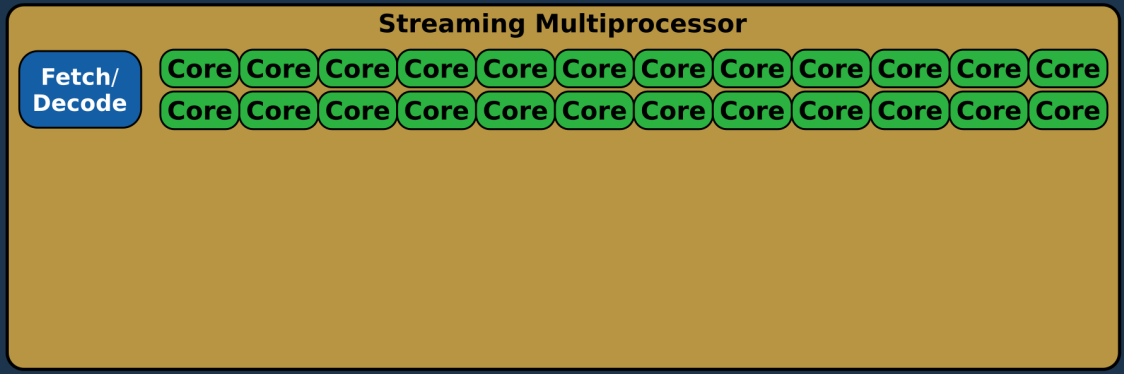
Streaming Multiprocessor



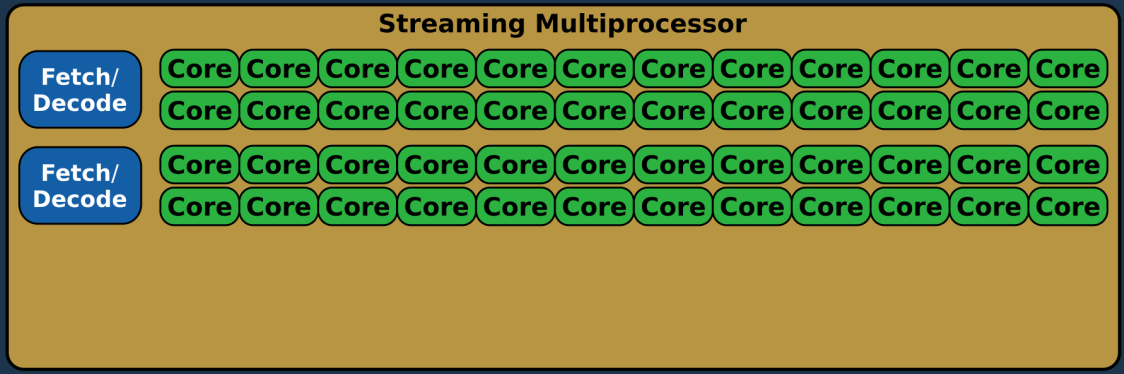
Streaming Multiprocessor



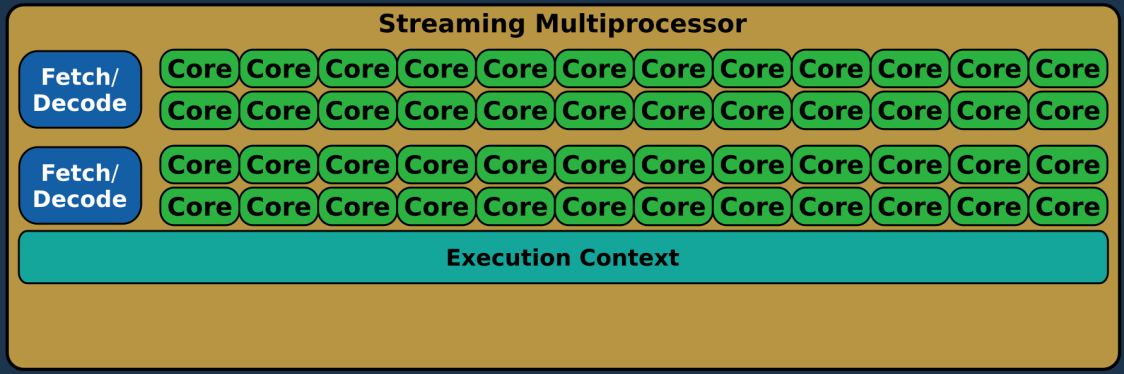
Streaming Multiprocessor



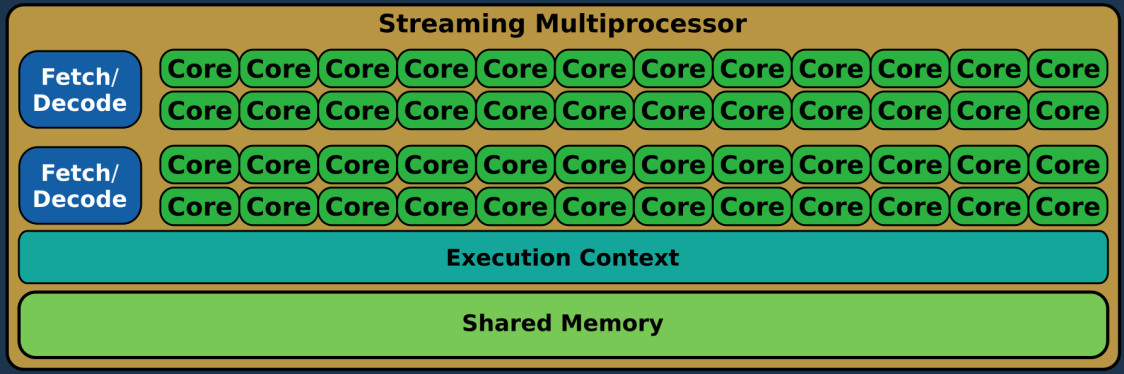
Streaming Multiprocessor



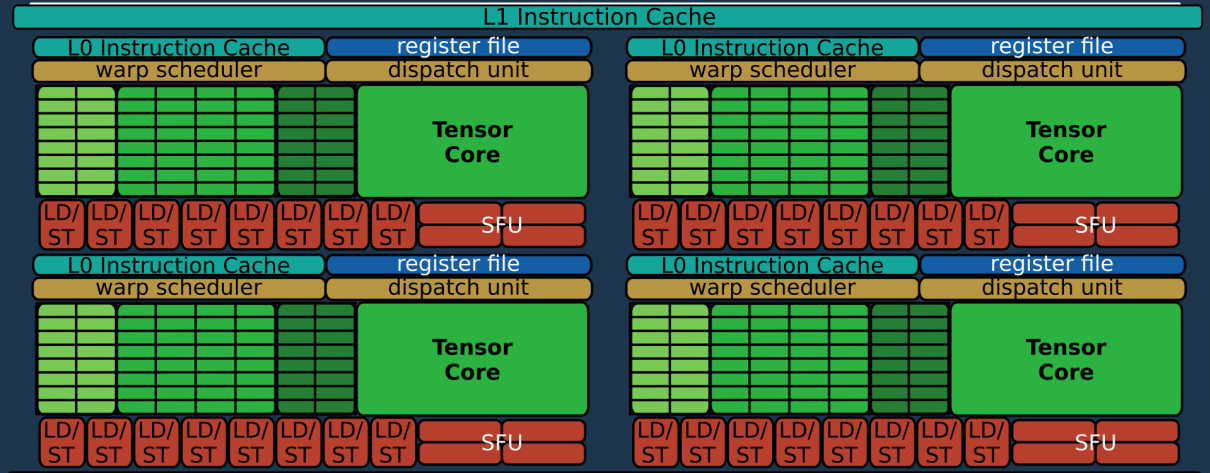
Streaming Multiprocessor



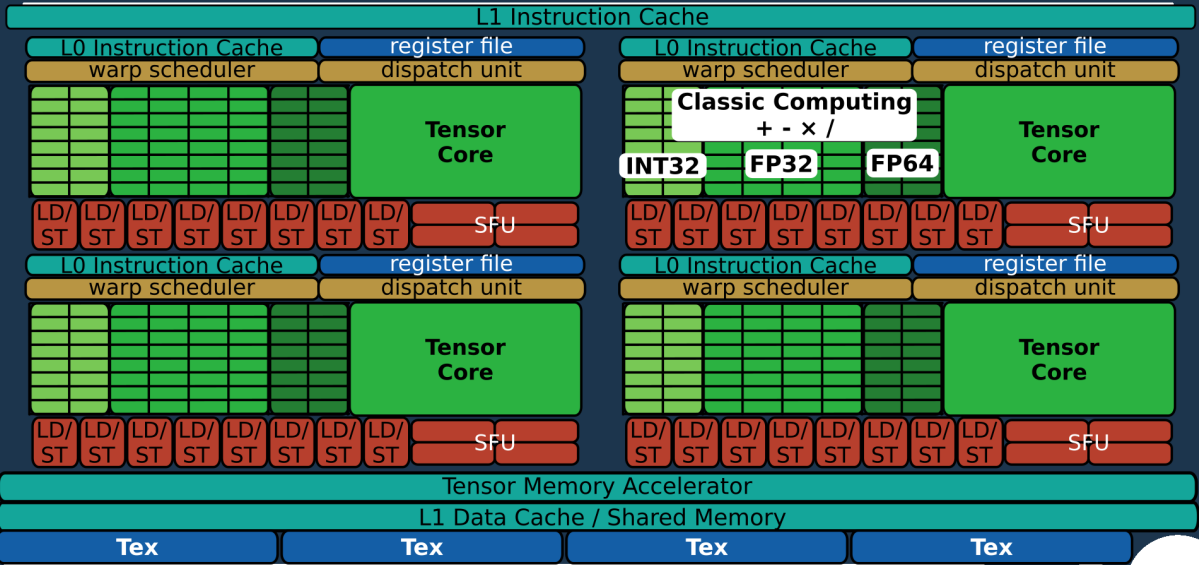
Streaming Multiprocessor



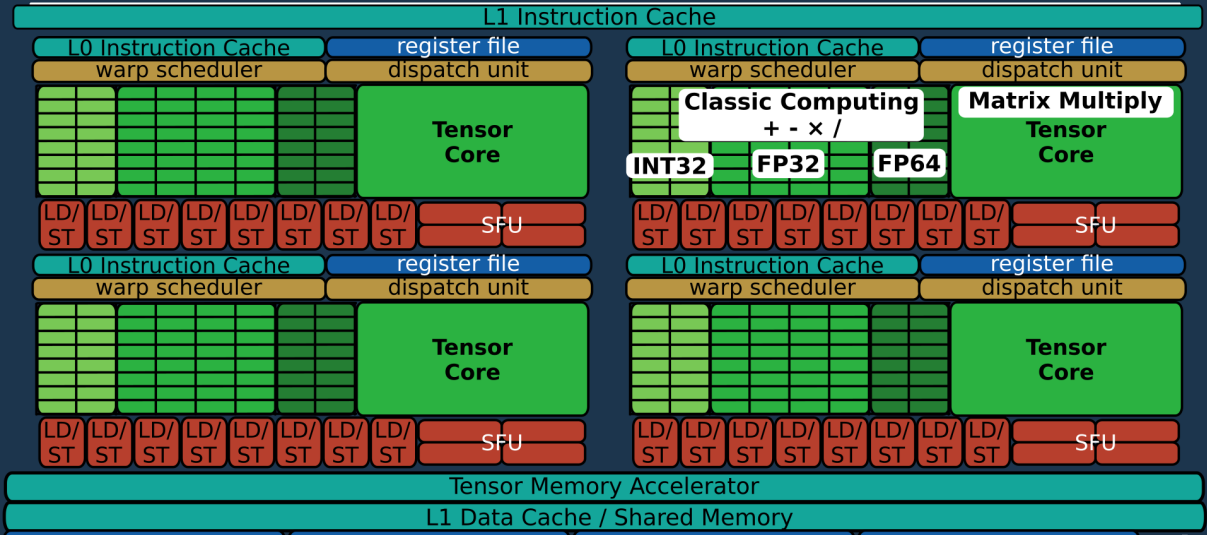
Streaming Multiprocessor H100



Streaming Multiprocessor H100

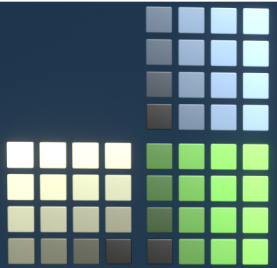


Streaming Multiprocessor H100

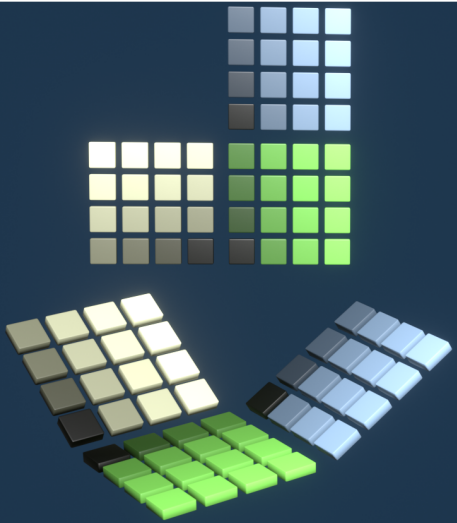


Tensor Core

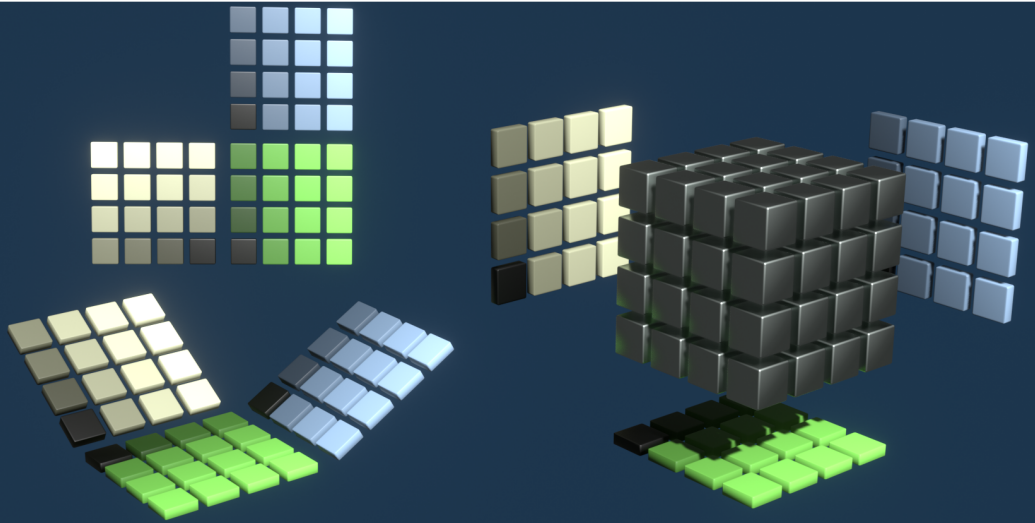
Tensor Core



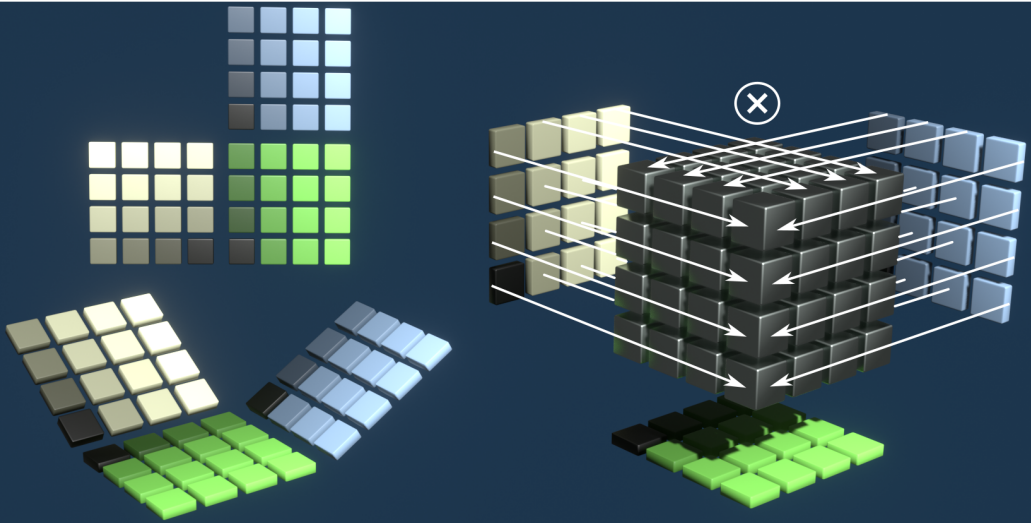
Tensor Core



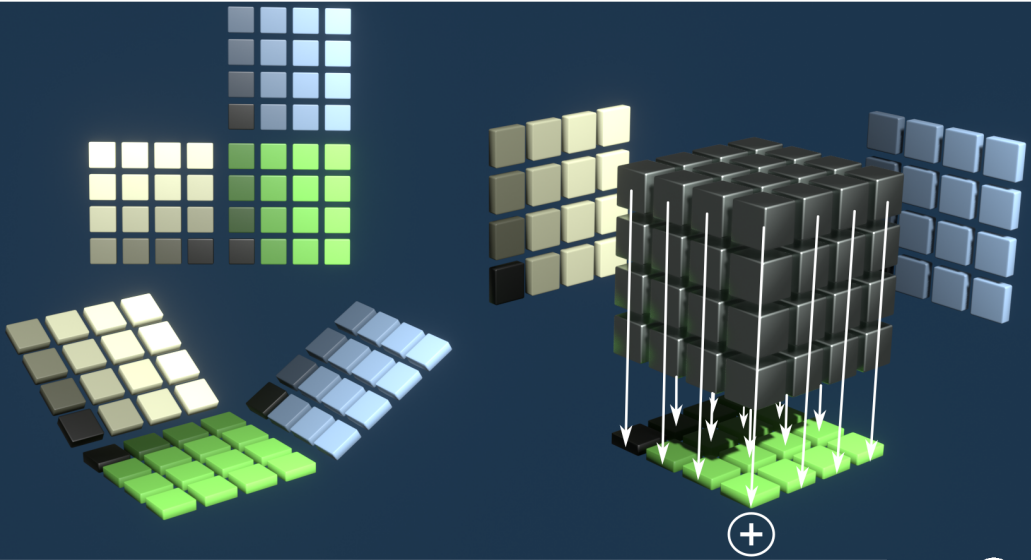
Tensor Core



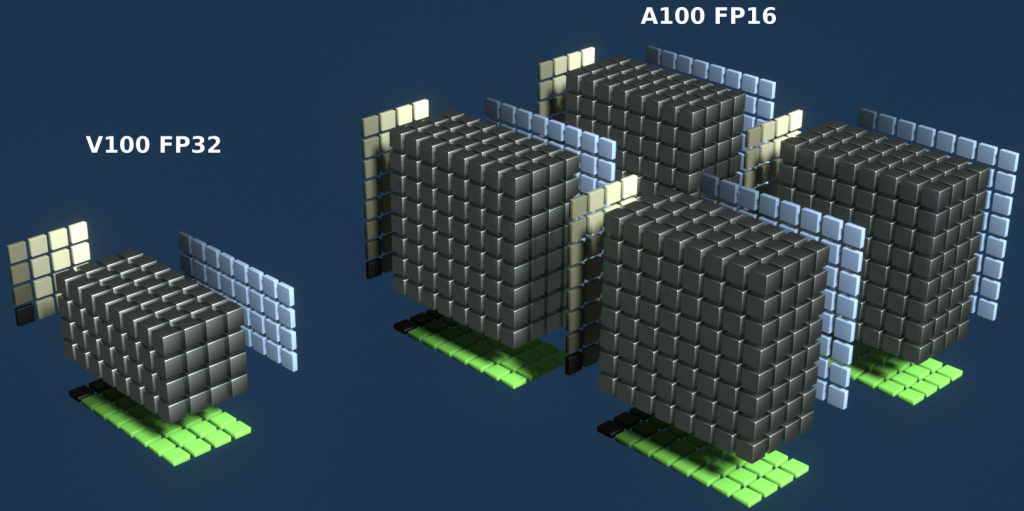
Tensor Core



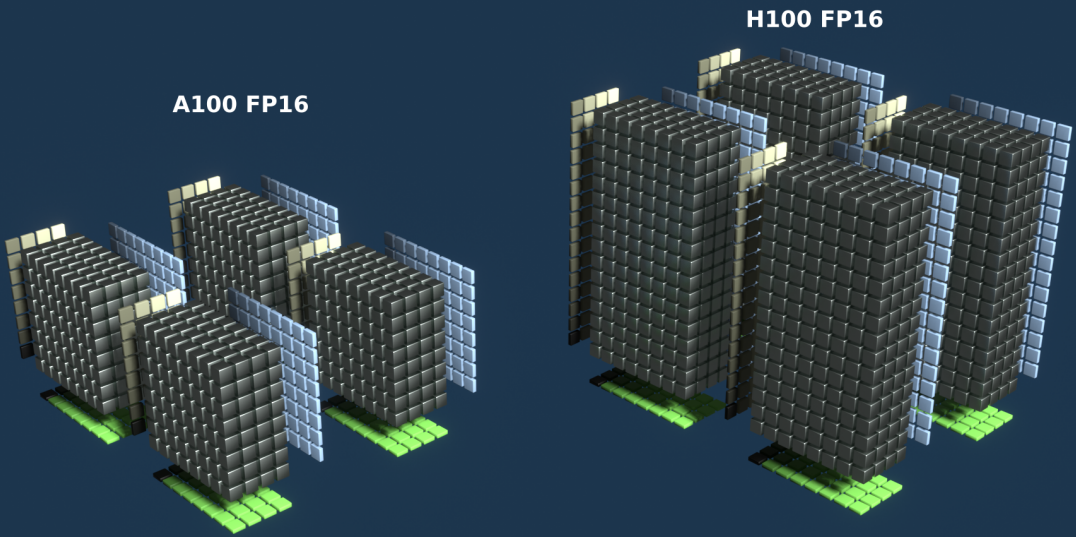
Tensor Core



Tensor Core

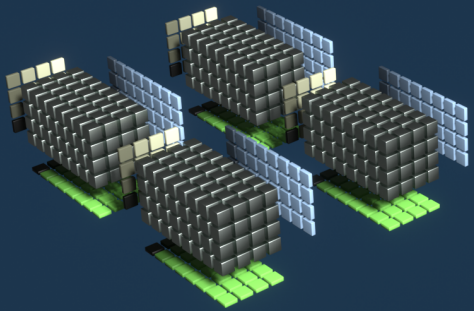


Tensor Core

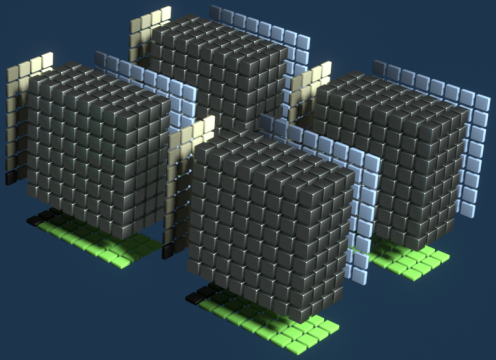


Tensor Core

A100 TF32



H100 TF32



Tensor Core

Tensor Core

Turring FP16

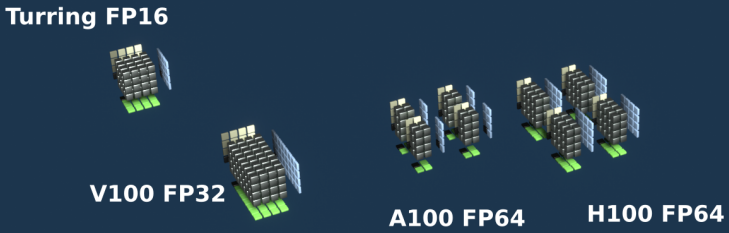
Tensor Core

Turring FP16

V100 FP32

Tensor Core

Tensor Core

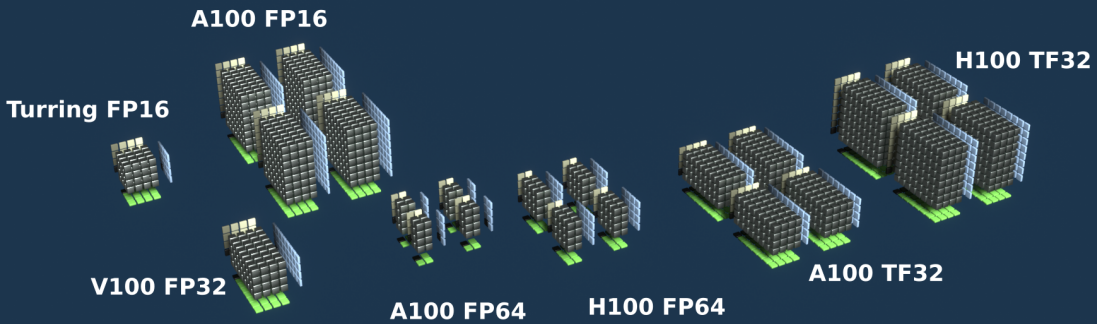


Tensor Core

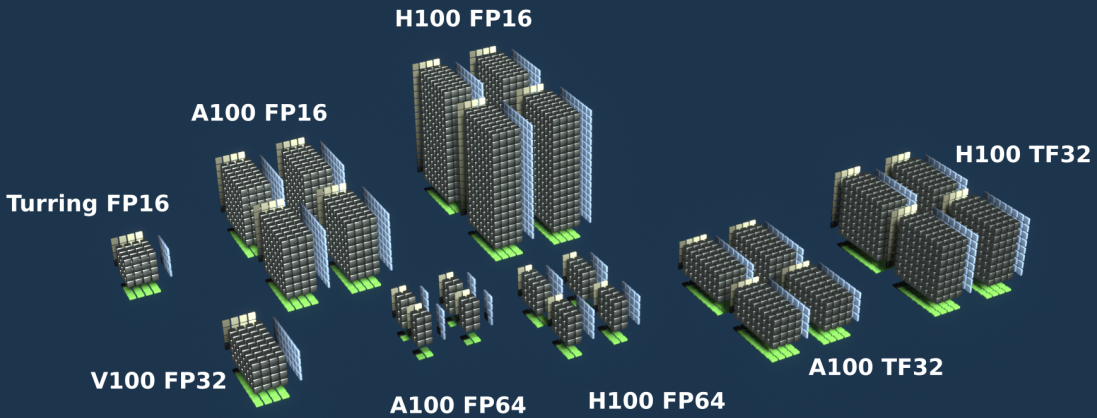
Tensor Core


Turring FP16
V100 FP32
A100 FP64
H100 FP64
A100 TF32
H100 TF32

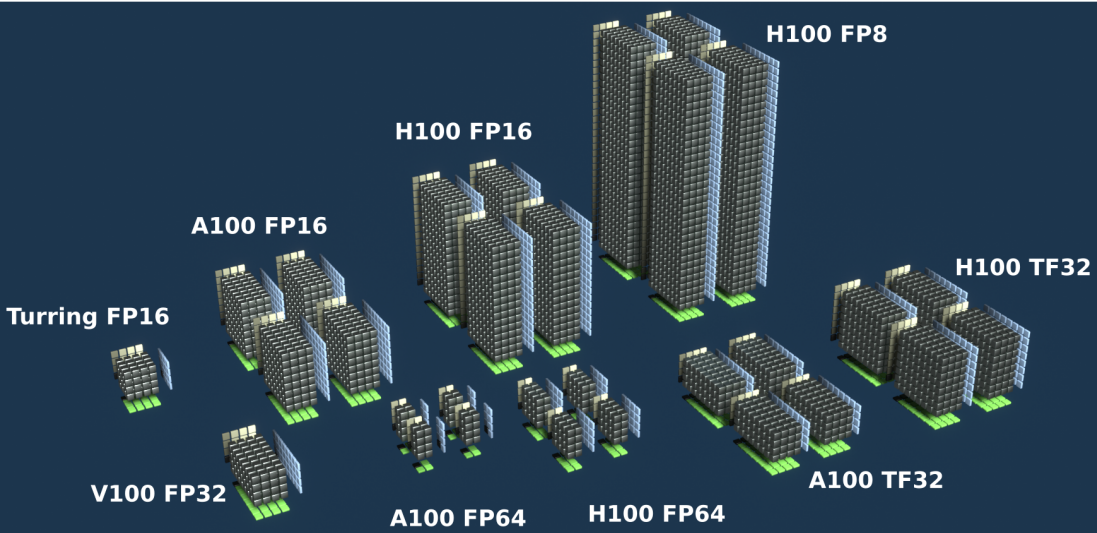
Tensor Core


A100 FP16
Turring FP16
H100 TF32
V100 FP32
A100 FP64
H100 FP64
A100 TF32

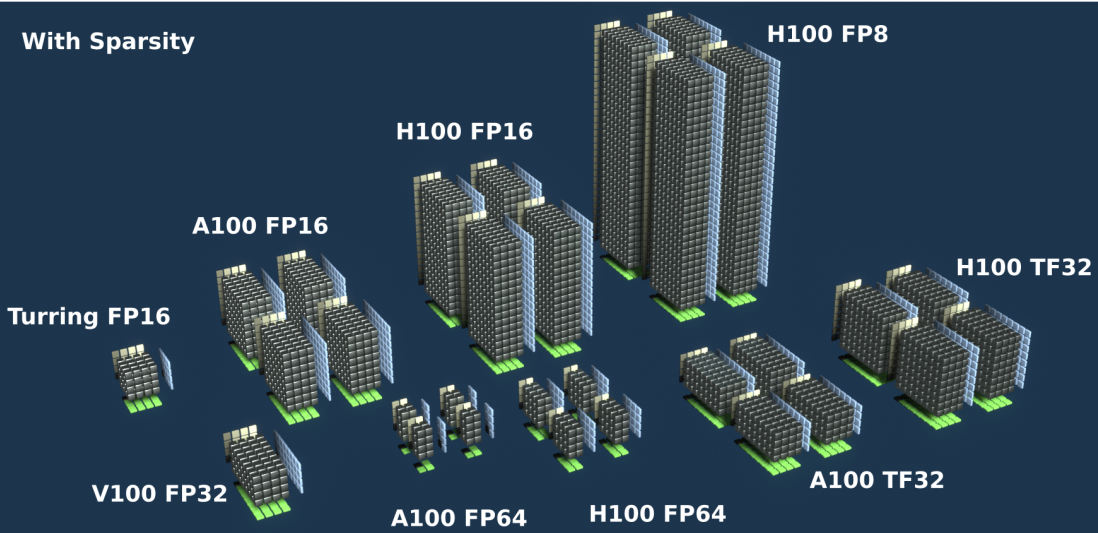
Tensor Core


H100 FP16
A100 FP16
H100 TF32
Turring FP16
V100 FP32
A100 FP64
H100 FP64
A100 TF32

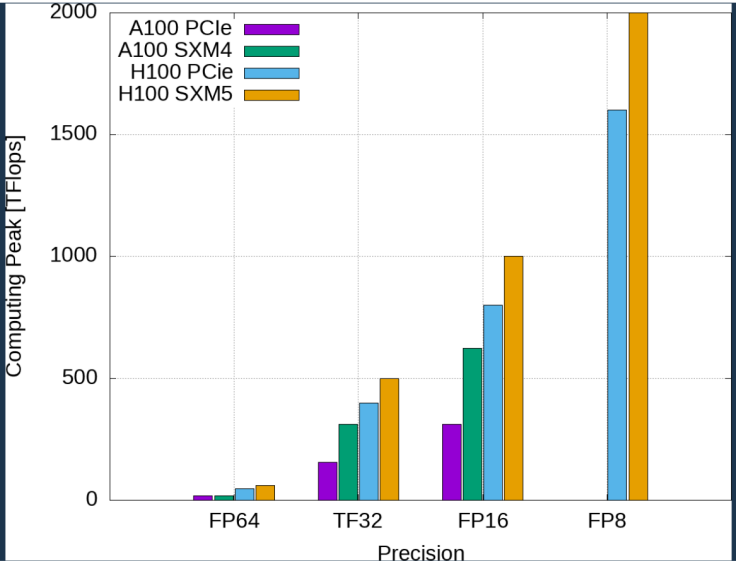
Tensor Core


H100 FP8
H100 FP16
A100 FP16
H100 TF32
Turring FP16
V100 FP32
A100 FP64
H100 FP64
A100 TF32

Tensor Core


With Sparsity
H100 FP8
H100 FP16
A100 FP16
H100 TF32
Turring FP16
V100 FP32
A100 FP64
H100 FP64
A100 TF32

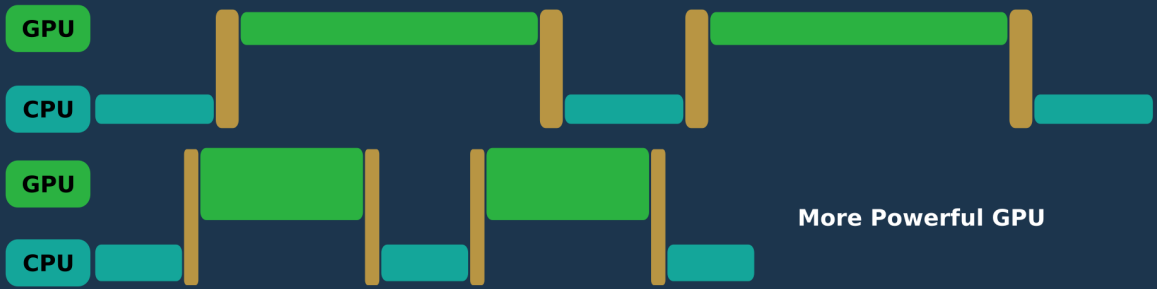
Tensor Core


A100 PCIe
A100 SXM4
H100 PCIe
H100 SXM5
Computing Peak [TFlops]
2000
1500
1000
500
0
FP64
TF32
FP16
FP8
Precision

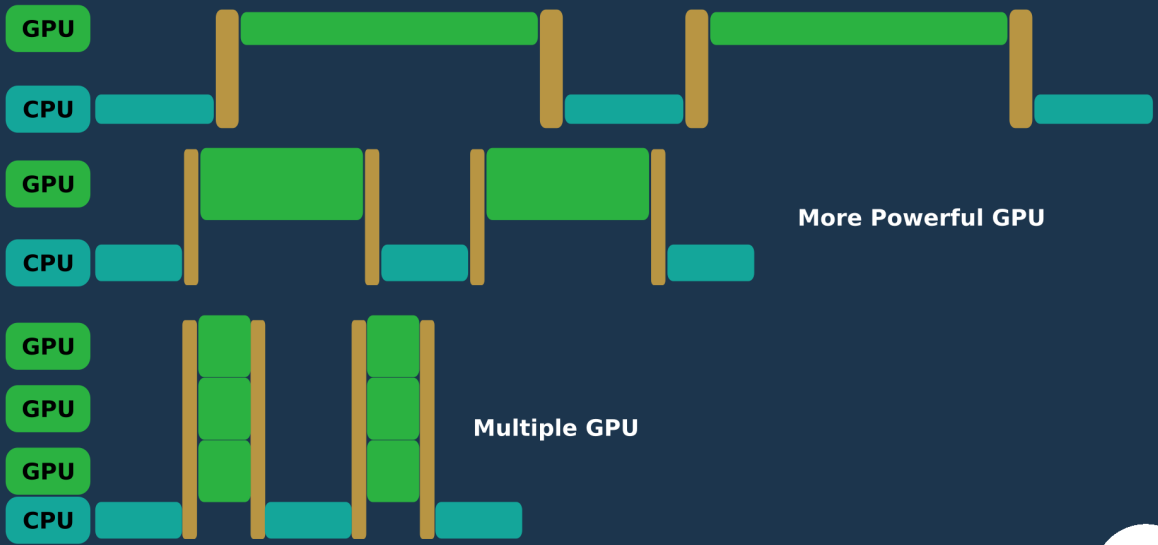
CPU + GPU Computing

CPU + GPU Computing

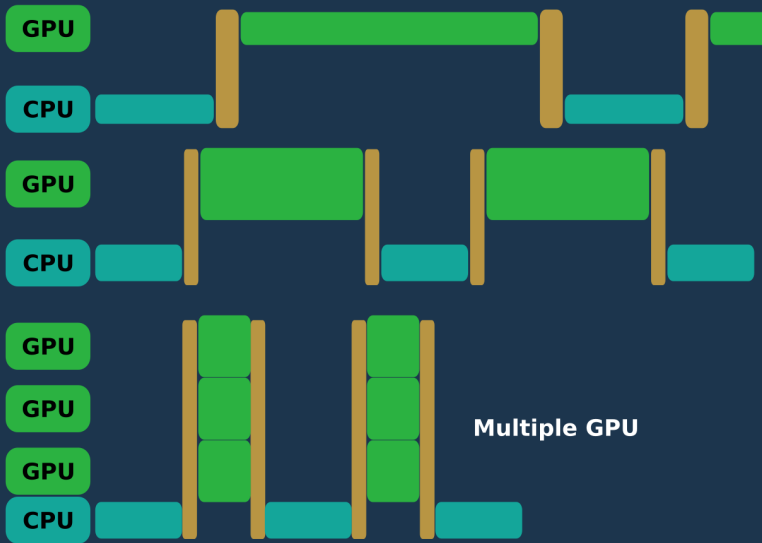
CPU + GPU Computing



CPU + GPU Computing



CPU + GPU Computing



More Powerful GPU

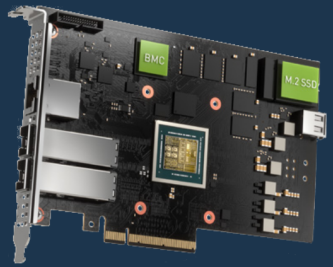
Stay on GPU as much as possible

Multiple GPU

Make less transfers as possible

Bluefield 3

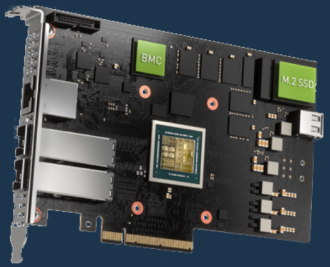
DPU : Data Processing Unit



Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

Bluefield 3

DPU : Data Processing Unit

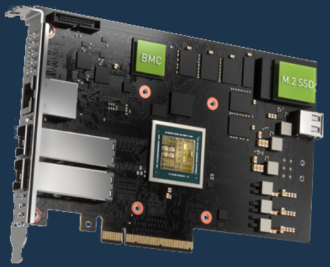


Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

1, 2, 4 ports with up to 400Gb/s connectivity
(Ethernet or NDR InfiniBand)

Bluefield 3

DPU : Data Processing Unit



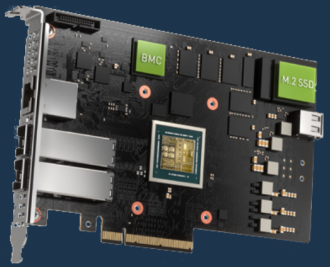
Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

1, 2, 4 ports with up to 400Gb/s connectivity
(Ethernet or NDR InfiniBand)

16 GB on-board DDR5 memory

Bluefield 3

DPU : Data Processing Unit



Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

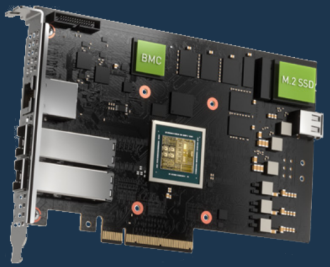
1, 2, 4 ports with up to 400Gb/s connectivity
(Ethernet or NDR InfiniBand)

16 GB on-board DDR5 memory

1 GbE out-of-band management port

Bluefield 3

DPU : Data Processing Unit



Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

1, 2, 4 ports with up to 400Gb/s connectivity
(Ethernet or NDR InfiniBand)

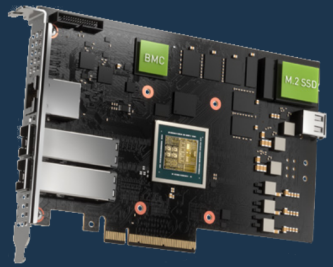
16 GB on-board DDR5 memory

1 GbE out-of-band management port

M.2 / U.2 connectors options for
direct attached storage

Bluefield 3

DPU : Data Processing Unit



Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

1, 2, 4 ports with up to 400Gb/s connectivity
(Ethernet or NDR InfiniBand)

16 GB on-board DDR5 memory

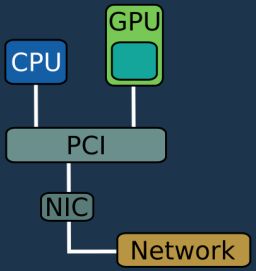
1 GbE out-of-band management port

M.2 / U.2 connectors options for
direct attached storage

Form factors: HHHL, FHHL

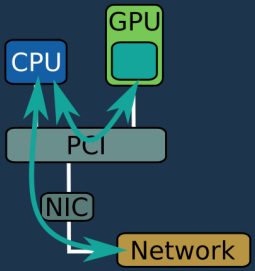
2022 - Hardware evolution

< 2022



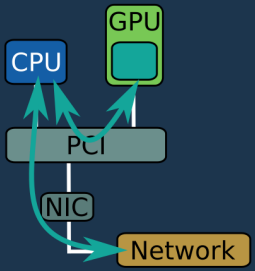
2022 - Hardware evolution

< 2022

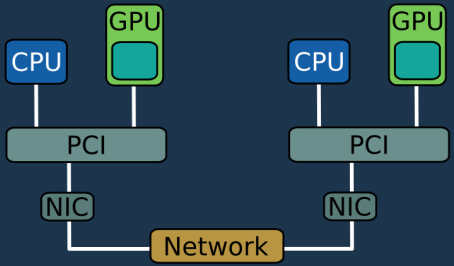


2022 - Hardware evolution

< 2022

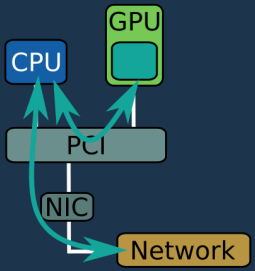


2022 and after

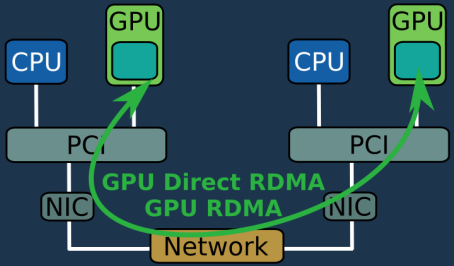


2022 - Hardware evolution

< 2022

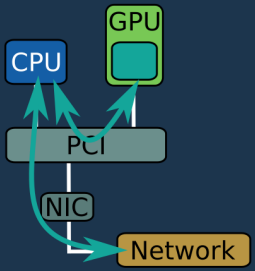


2022 and after

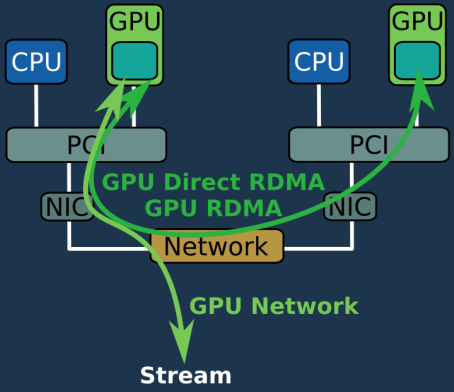


2022 - Hardware evolution

< 2022

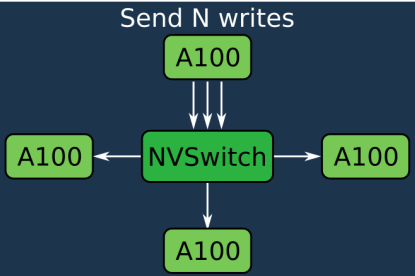


2022 and after

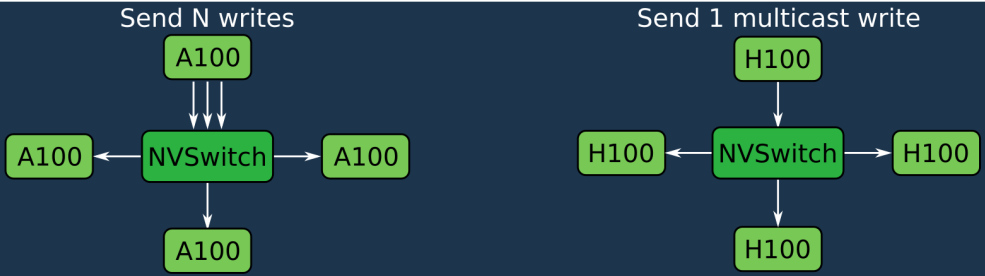


NVLink/NVSwitch acceleration

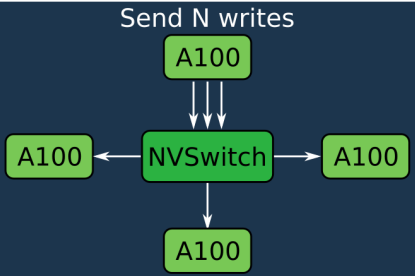
NVLink/NVSwitch acceleration

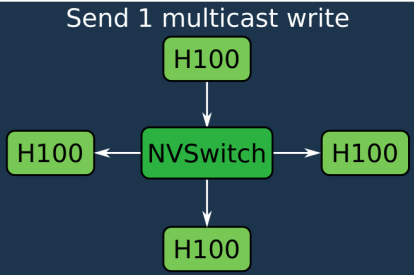


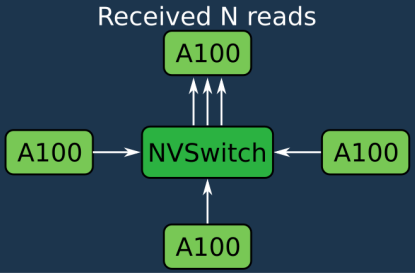
NVLink/NVSwitch acceleration



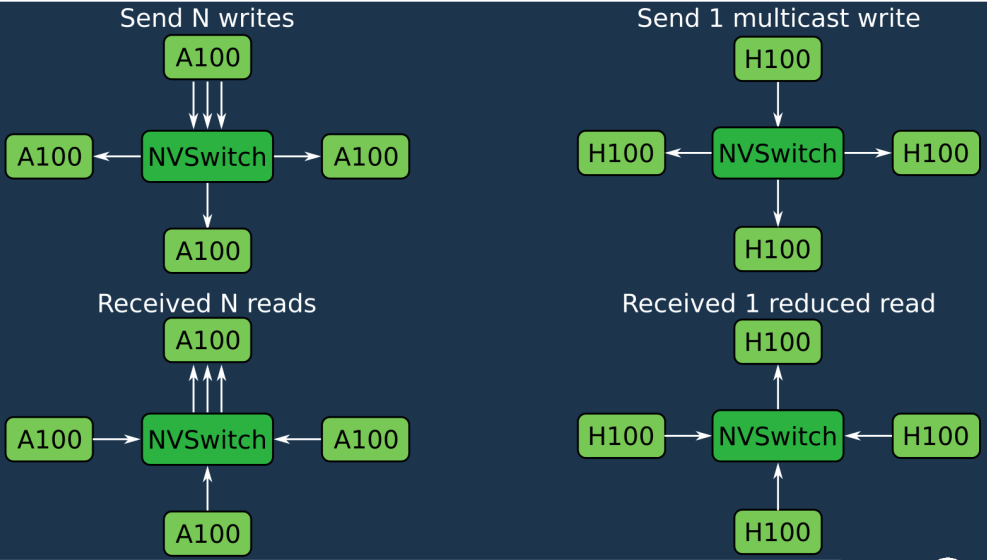
NVLink/NVSwitch acceleration





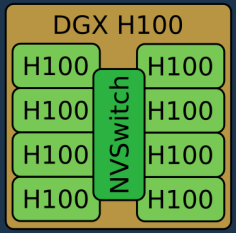


NVLink/NVSwitch acceleration

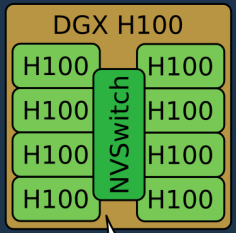


DGX H100 / DGX POD H100

DGX H100 / DGX POD H100

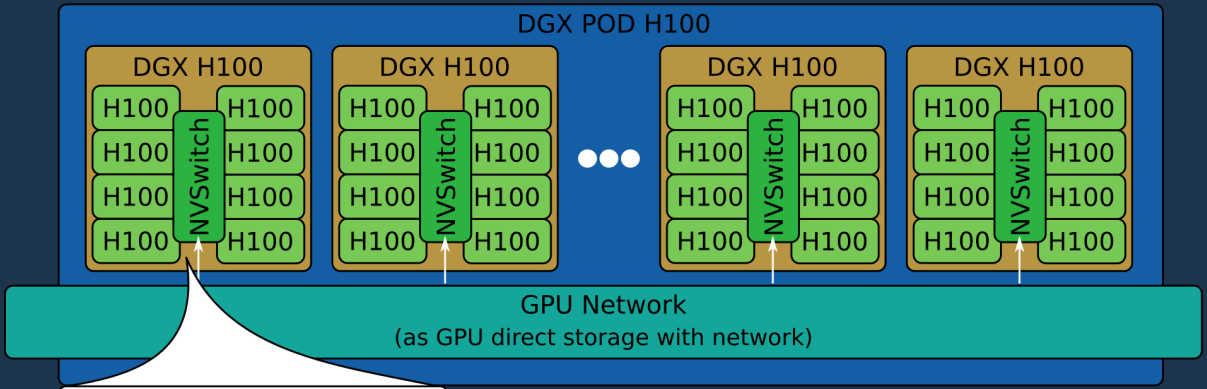


DGX H100 / DGX POD H100



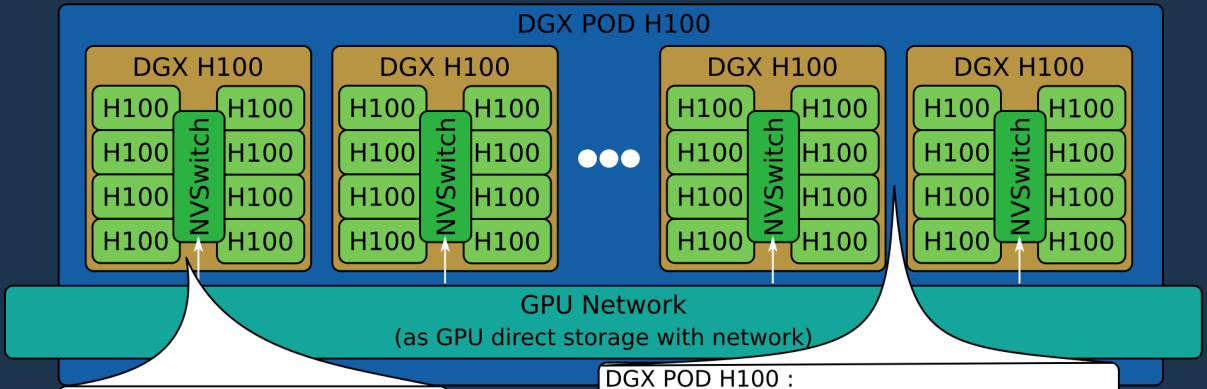
DGX H100 :
- 8 H100
- 32 PFlop (AI performance)
- 640 GB HBM3 Memory
- 24 TB/s memory Bandwidth

DGX H100 / DGX POD H100



DGX H100 :
- 8 H100
- 32 PFlop (AI performance)
- 640 GB HBM3 Memory
- 24 TB/s memory Bandwidth

DGX H100 / DGX POD H100

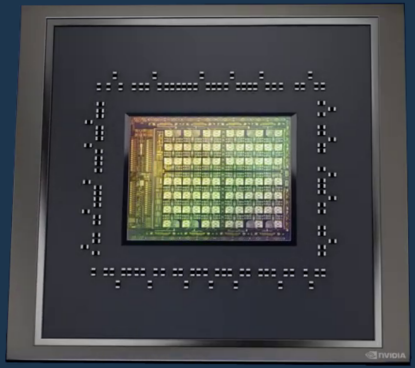


DGX H100 :
- 8 H100
- 32 PFlop (AI performance)
- 640 GB HBM3 Memory
- 24 TB/s memory Bandwidth

DGX POD H100 :
- 32 DGX H100
- 1 EFlop (AI performance)
- 20 TB HBM3 Memory
- 70 TB/s memory Bandwidth
- 192 TFlops of Sharp in-Network compute

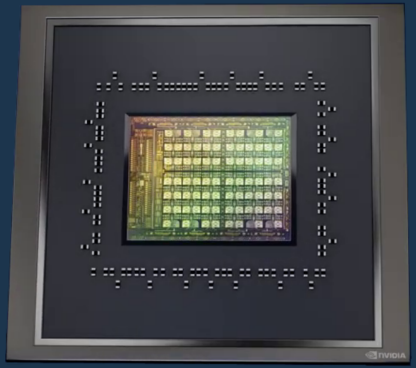
2023 Grace CPU

Grace



2023 Grace CPU

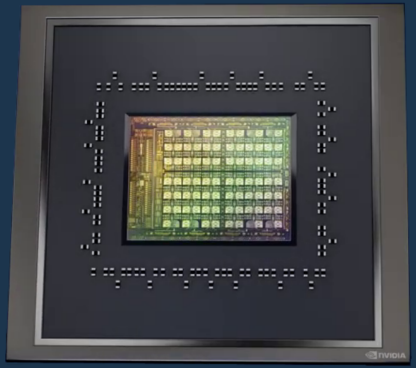
Grace



72 Neoverse V2 cores, Armv9
SVE2 with 4x128b

2023 Grace CPU

Grace

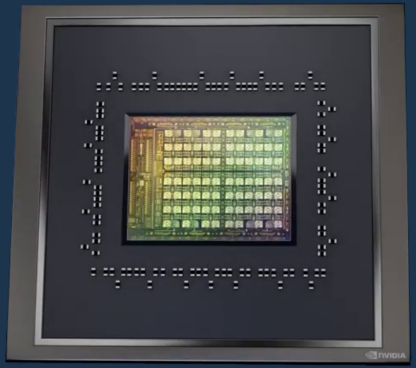


72 Neoverse V2 cores, Armv9 SVE2 with 4x128b

240 GB RAM (LPDDR5X)

2023 Grace CPU

Grace



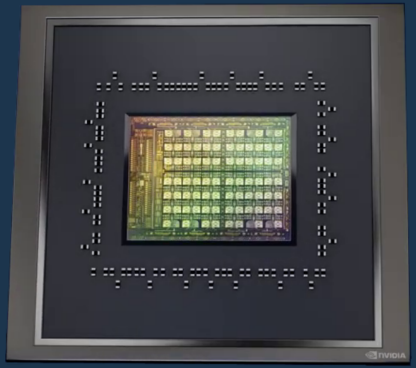
72 Neoverse V2 cores, Armv9 SVE2 with 4x128b

240 GB RAM (LPDDR5X)

3.2 TB/s Bi-section BW

2023 Grace CPU

Grace



72 Neoverse V2 cores, Armv9 SVE2 with 4x128b

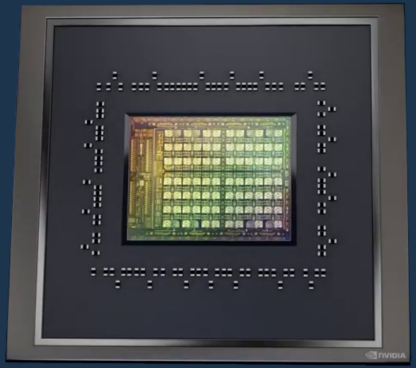
240 GB RAM (LPDDR5X)

3.2 TB/s Bi-section BW

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 117 MB

2023 Grace CPU

Grace



72 Neoverse V2 cores, Armv9
SVE2 with 4x128b

240 GB RAM (LPDDR5X)

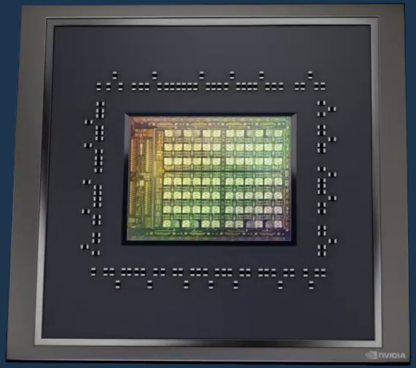
3.2 TB/s Bi-section BW

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 117 MB

Local caching of remote CPU/GPU memory
Supports up to 4 chip coherency over Coherent NVLINK

2023 Grace CPU

Grace



72 Neoverse V2 cores, Armv9
SVE2 with 4x128b

240 GB RAM (LPDDR5X)

3.2 TB/s Bi-section BW

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 117 MB

Local caching of remote CPU/GPU memory
Supports up to 4 chip coherency over Coherent NVLINK

NVLink C2C (900 GB/s)

2023 - Grace Superchip

Grace Superchip

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

1 TB/s Bandwidth (HBM3)

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

1 TB/s Bandwidth (HBM3)

960 GB RAM (LPDDR5X)

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

1 TB/s Bandwidth (HBM3)

960 GB RAM (LPDDR5X)

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 234 MB per superchip

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

1 TB/s Bandwidth (HBM3)

960 GB RAM (LPDDR5X)

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 234 MB per superchip

NVLink Gen 4 (900 GB/s)

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

1 TB/s Bandwidth (HBM3)

960 GB RAM (LPDDR5X)

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 234 MB per superchip

NVLink Gen 4 (900 GB/s)

7.1 TFlops (FP64 computing peak)

2023 - Grace Superchip

Grace Superchip

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

1 TB/s Bandwidth (HBM3)

960 GB RAM (LPDDR5X)

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 234 MB per superchip

NVLink Gen 4 (900 GB/s)

7.1 TFlops (FP64 computing peak)

500 W

2023 - Grace Superchip

Grace Superchip

Passive air colling

Neoverse V2 Cores: Armv9 with 4x128b SVE2

144 Cores

1 TB/s Bandwidth (HBM3)

960 GB RAM (LPDDR5X)

Cache :
- L1: 64 KB (I/D) per core
- L2: 1 MB per core
- L3: 234 MB per superchip

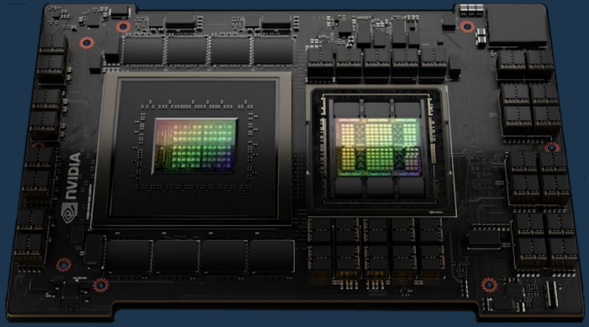
NVLink Gen 4 (900 GB/s)

7.1 TFlops (FP64 computing peak)

500 W

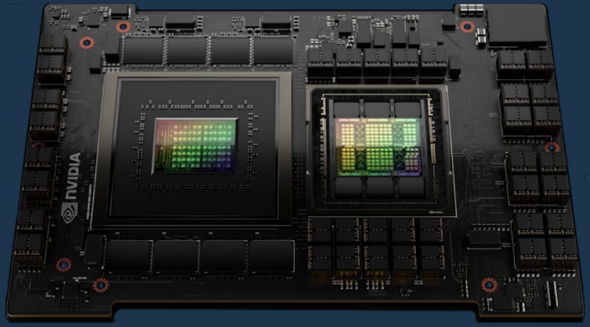
2023 - Grace Hopper

Grace Hopper Superchip



2023 - Grace Hopper

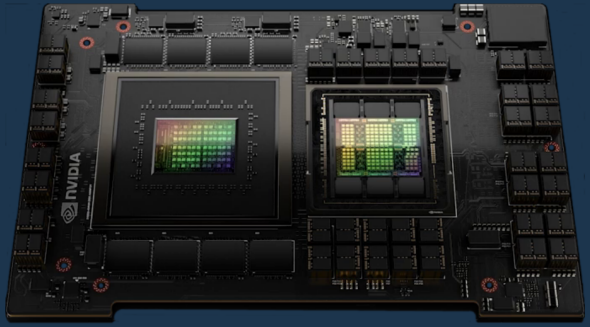
Grace Hopper Superchip



4 TB/s Bandwidth

2023 - Grace Hopper

Grace Hopper Superchip

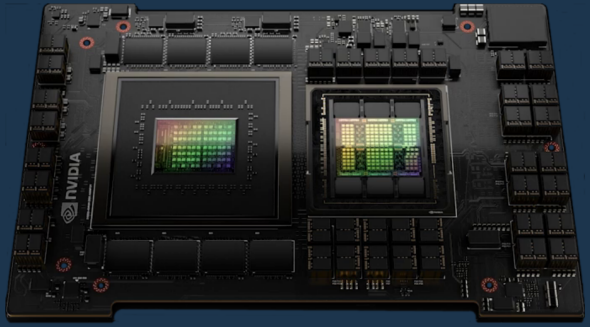


4 TB/s Bandwidth

600 GB RAM (LPDDR5X)

2023 - Grace Hopper

Grace Hopper Superchip



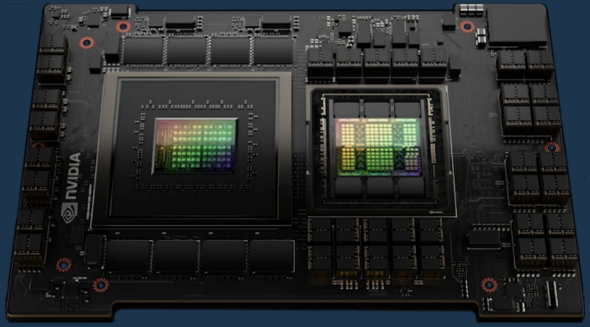
4 TB/s Bandwidth

600 GB RAM (LPDDR5X)

NVLink C2C (900 GB/s)

2023 - Grace Hopper

Grace Hopper Superchip



4 TB/s Bandwidth

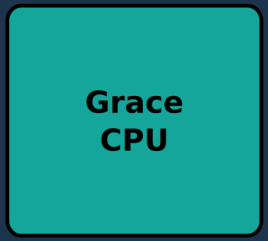
600 GB RAM (LPDDR5X)

NVLink C2C (900 GB/s)

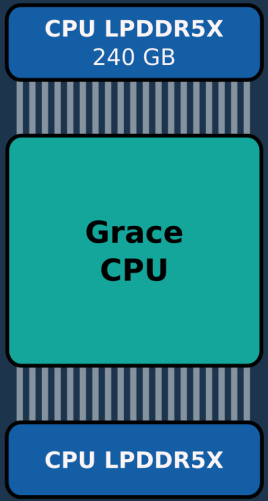
Unified Memory

Grace Hopper Superchip

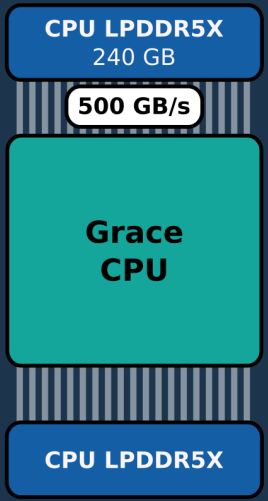
Grace Hopper Superchip



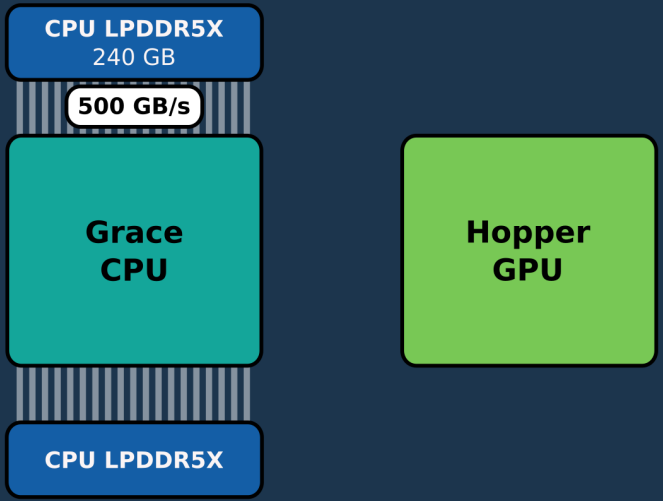
Grace Hopper Superchip



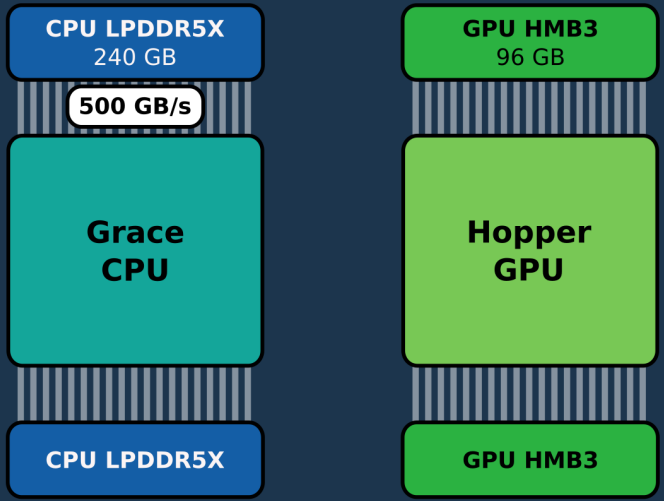
Grace Hopper Superchip



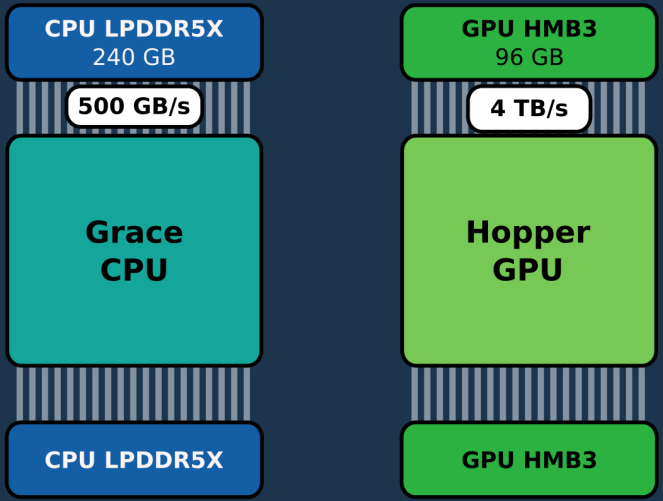
Grace Hopper Superchip



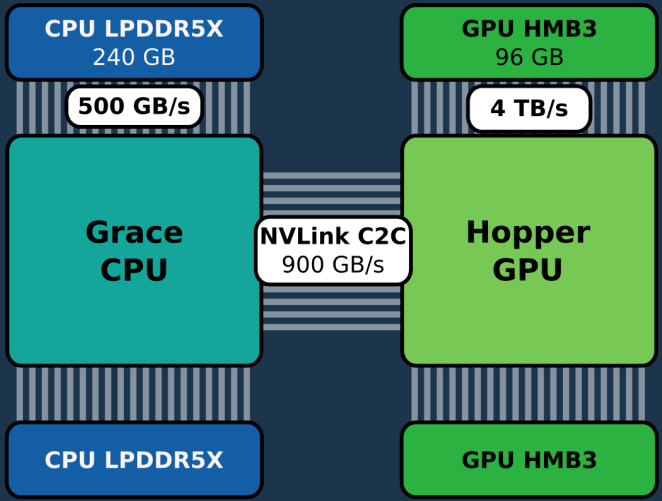
Grace Hopper Superchip



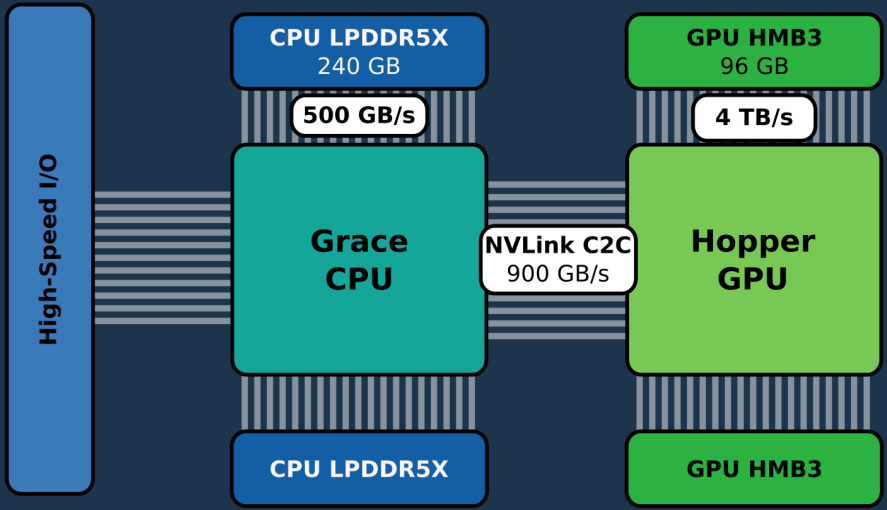
Grace Hopper Superchip



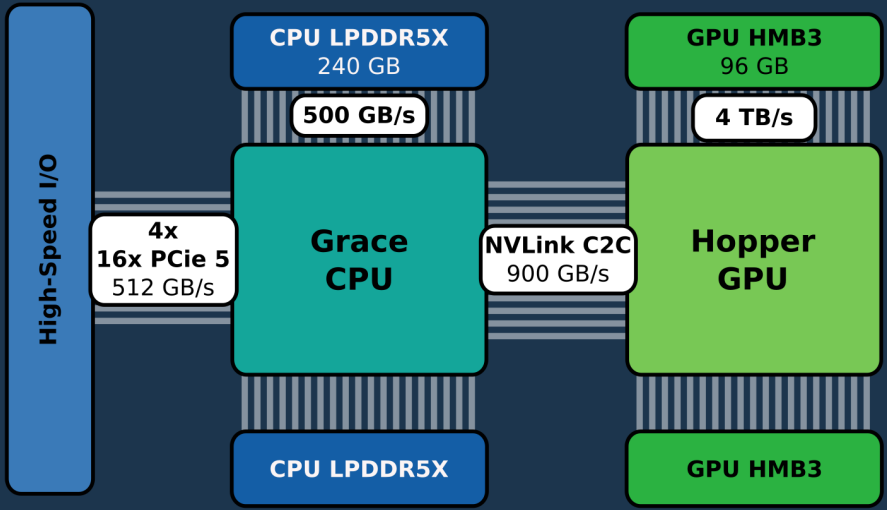
Grace Hopper Superchip



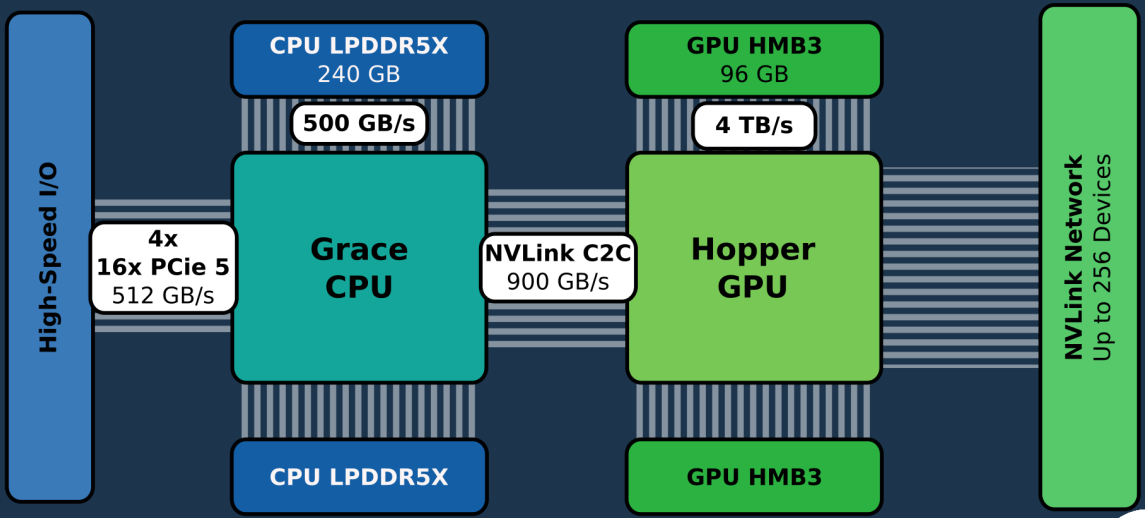
Grace Hopper Superchip



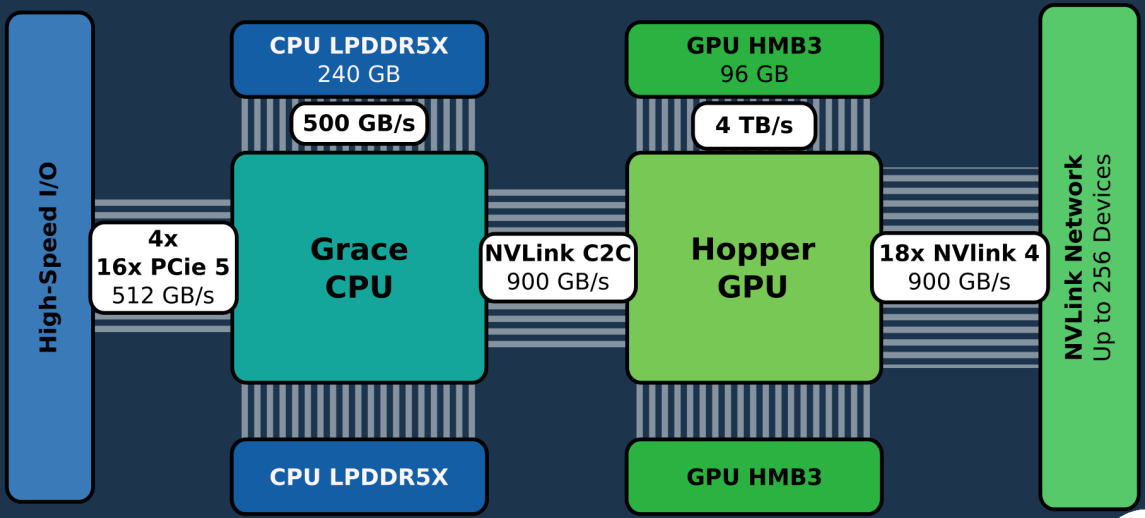
Grace Hopper Superchip



Grace Hopper Superchip



Grace Hopper Superchip

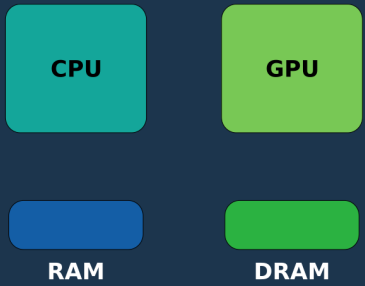


Memory Management

Memory Management

CUDA

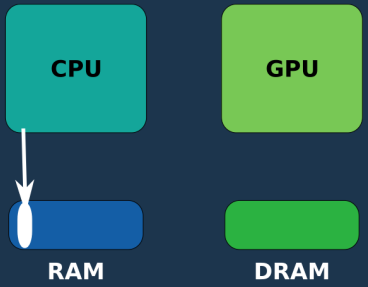
< 2020



Memory Management

CUDA

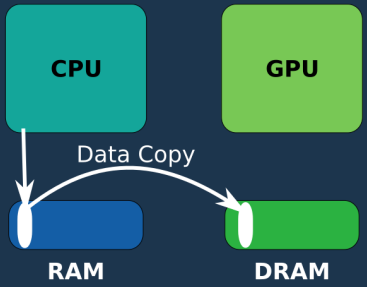
< 2020



Memory Management

CUDA

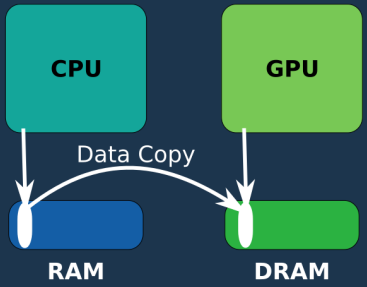
< 2020



Memory Management

CUDA

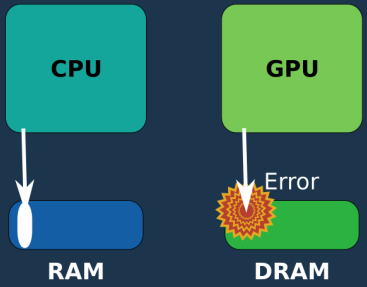
< 2020



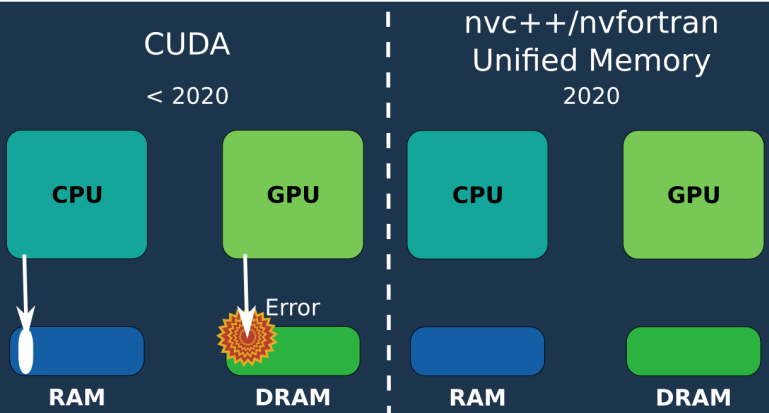
Memory Management

CUDA

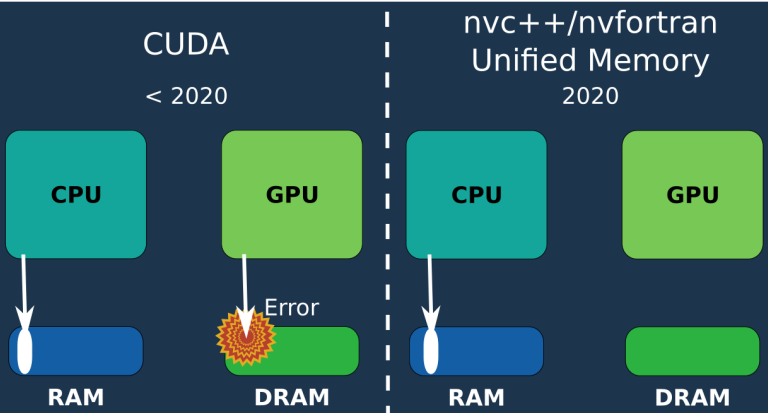
< 2020



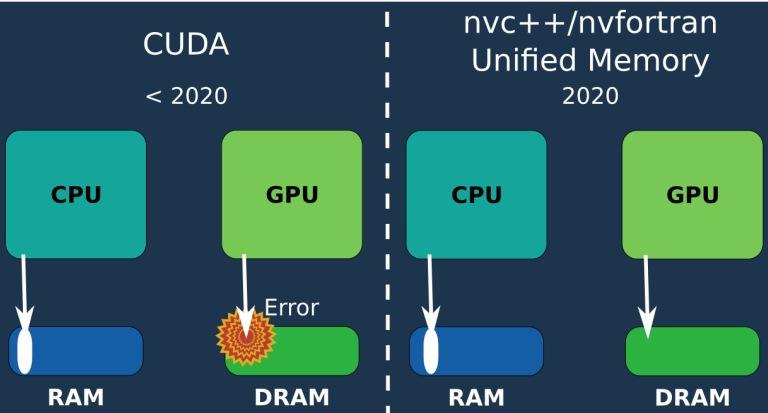
Memory Management



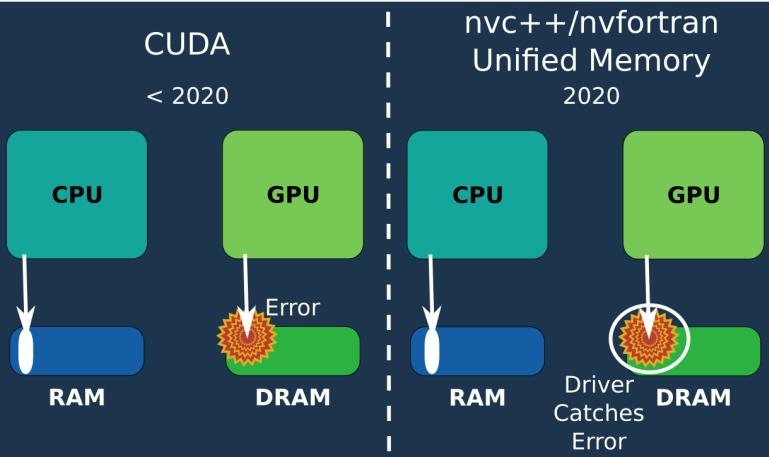
Memory Management



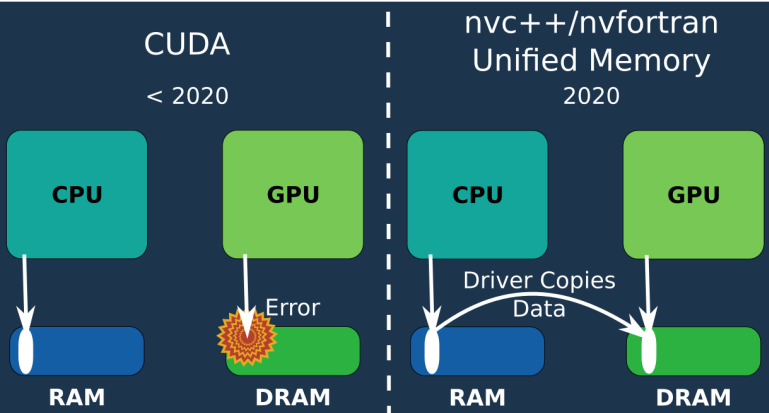
Memory Management



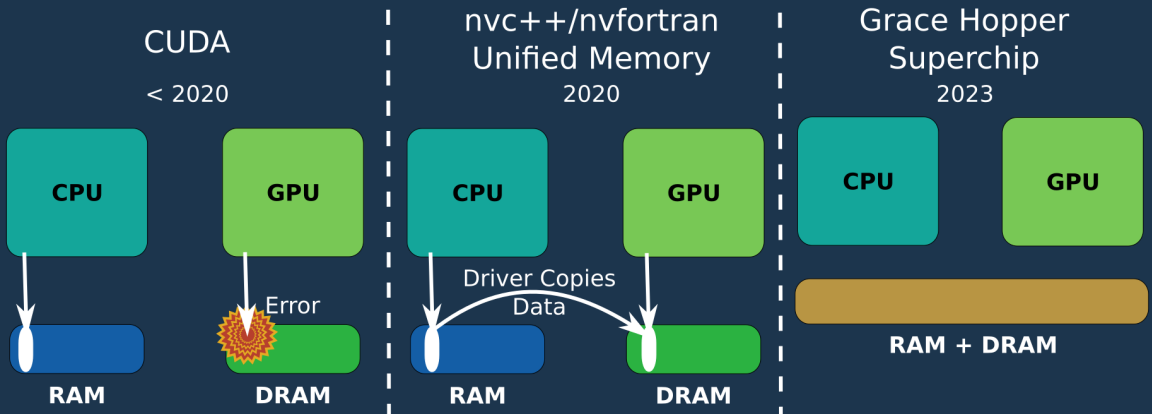
Memory Management



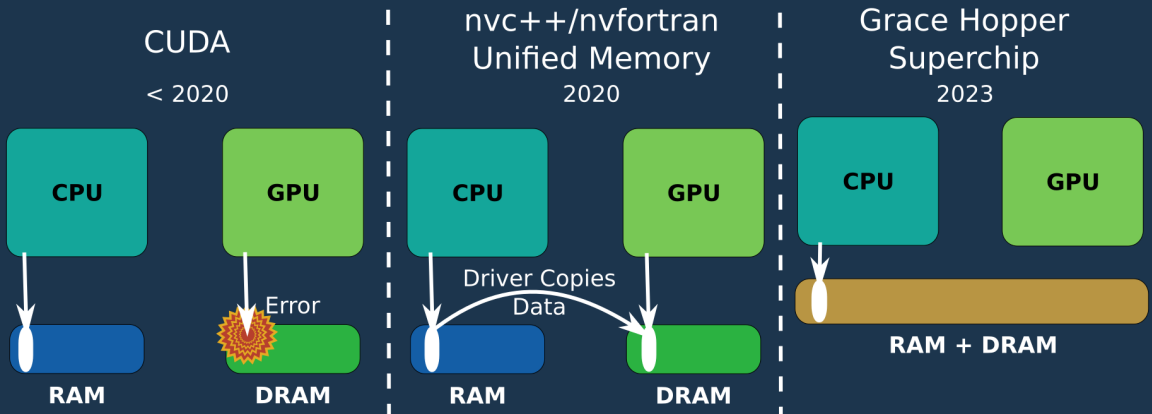
Memory Management



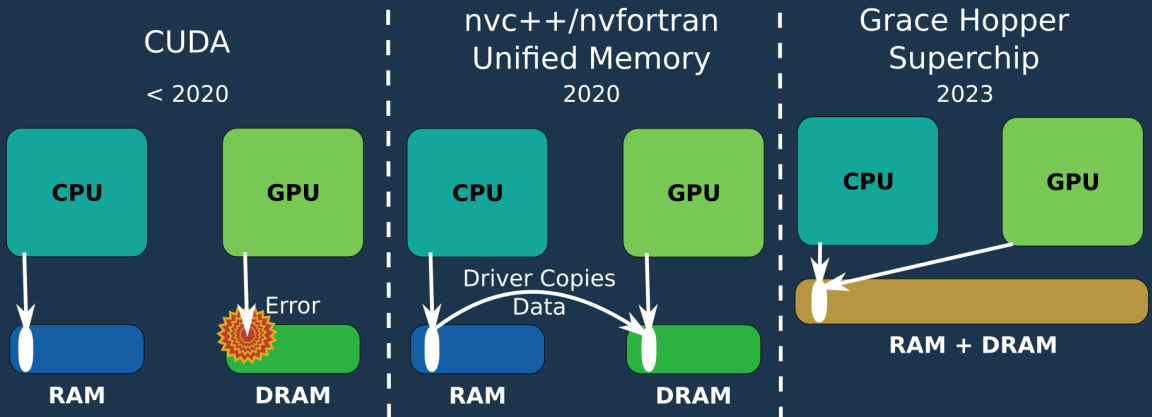
Memory Management



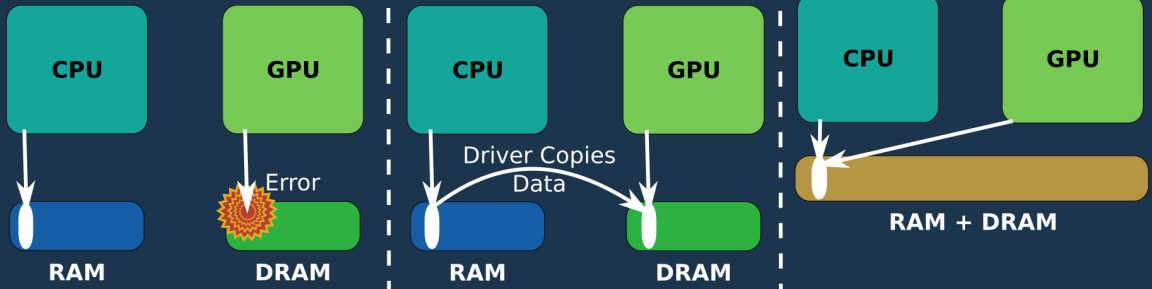
Memory Management



Memory Management

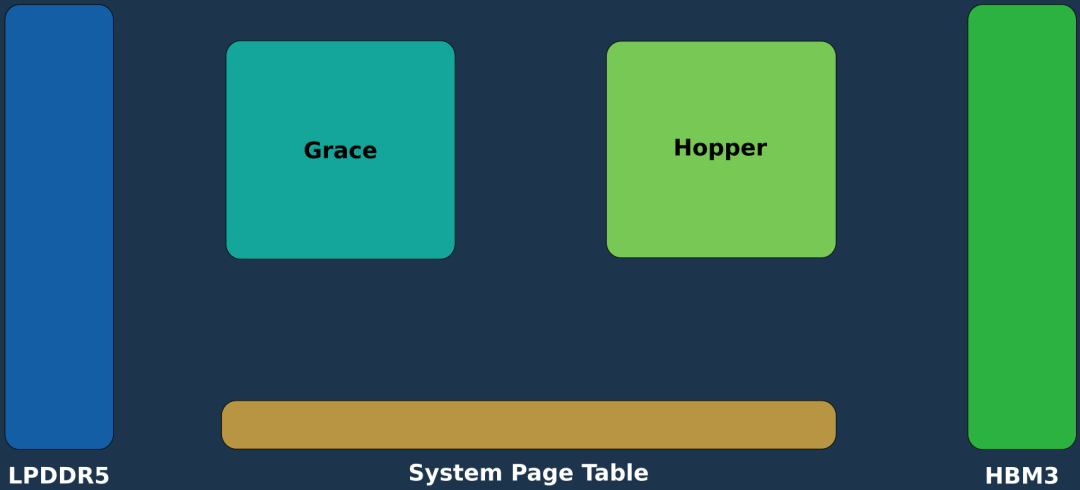


Memory Management

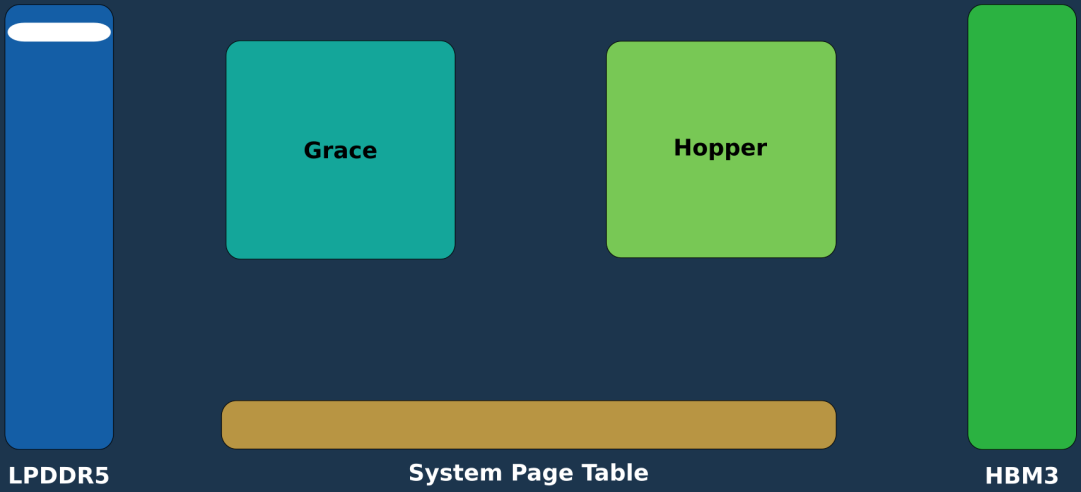


Yes but **RAM** and **DRAM** are **physically different**

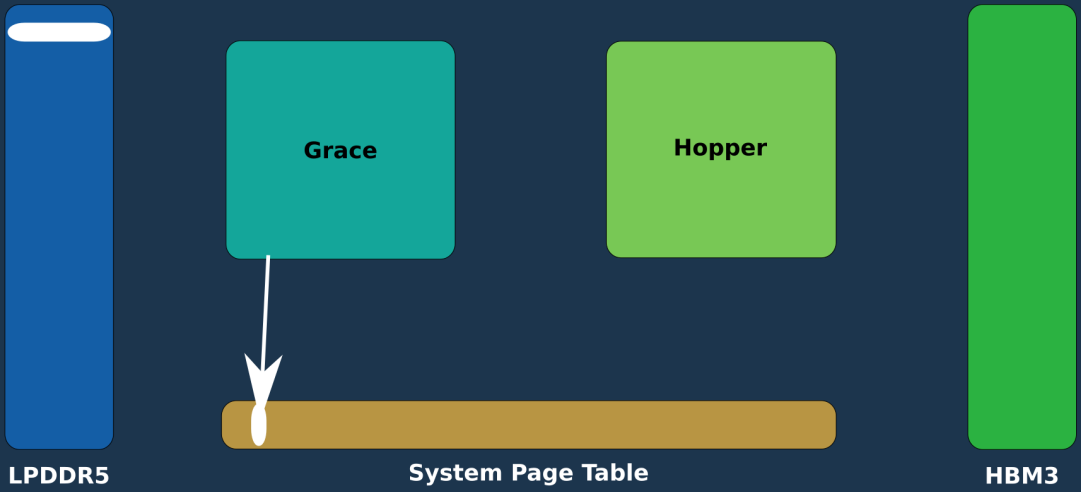
Memory Management Grace Hopper



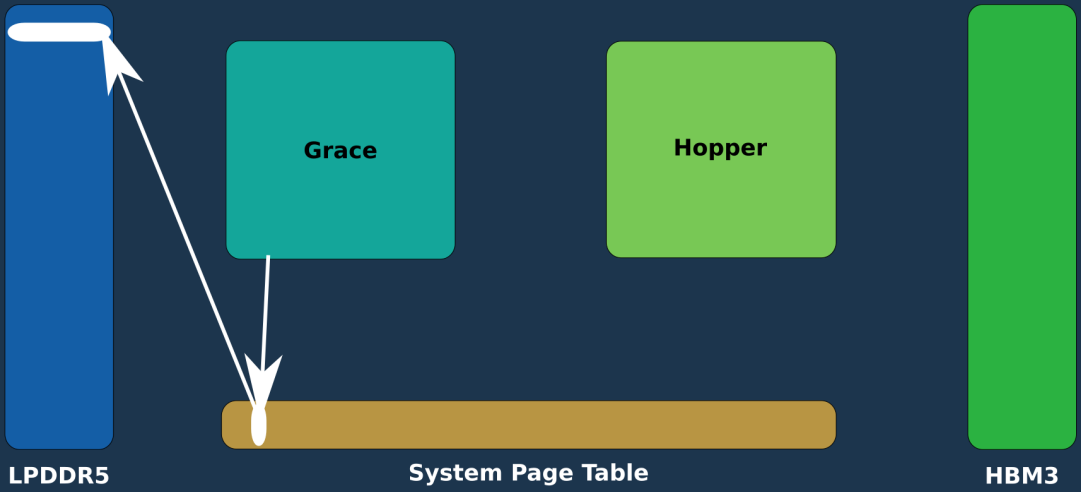
Memory Management Grace Hopper



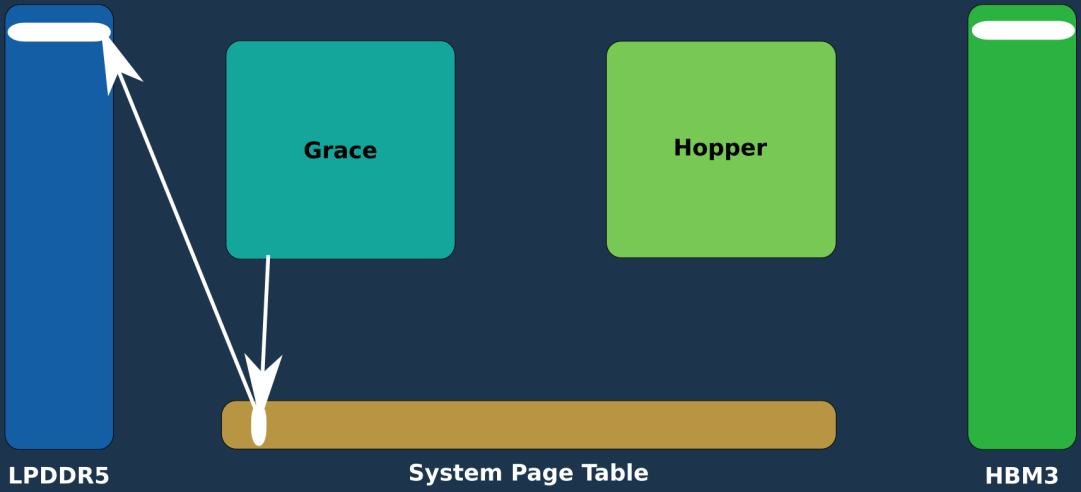
Memory Management Grace Hopper



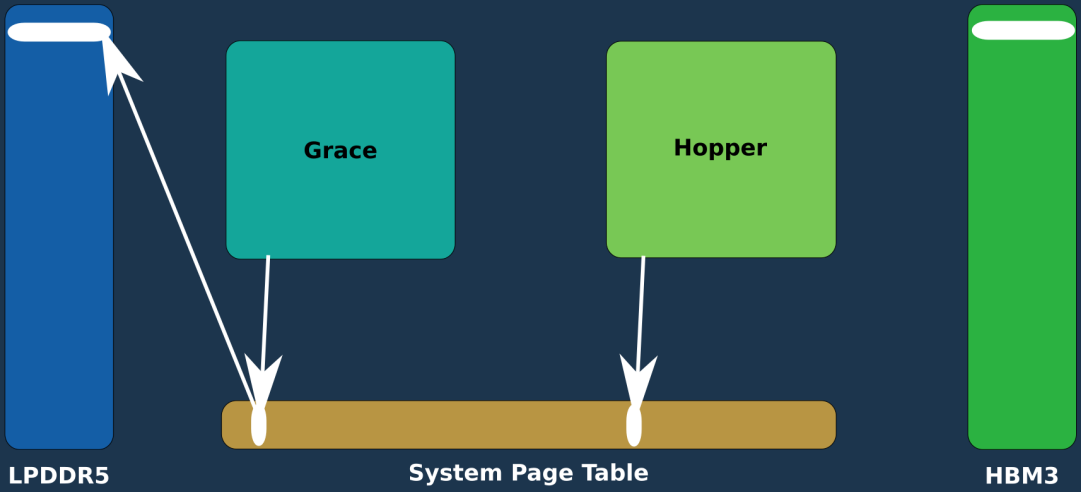
Memory Management Grace Hopper



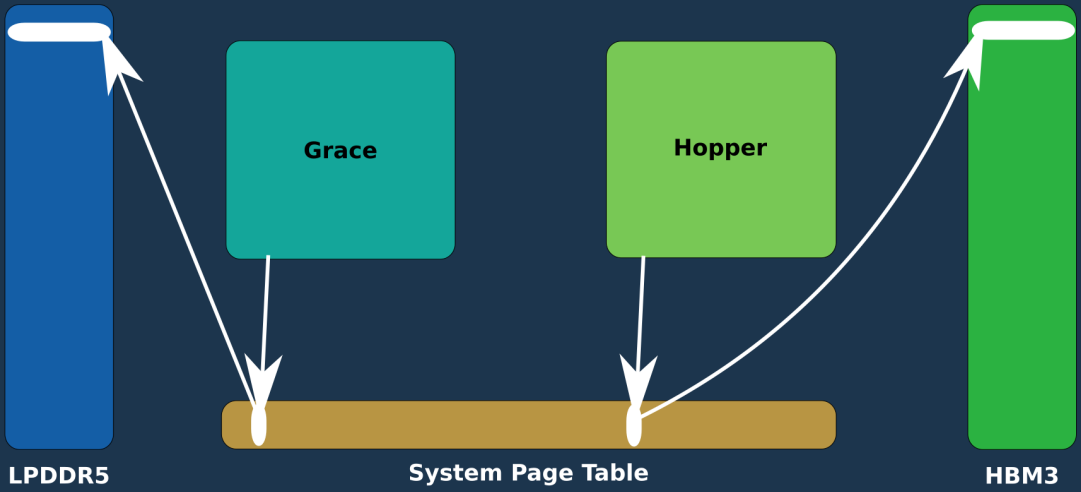
Memory Management Grace Hopper



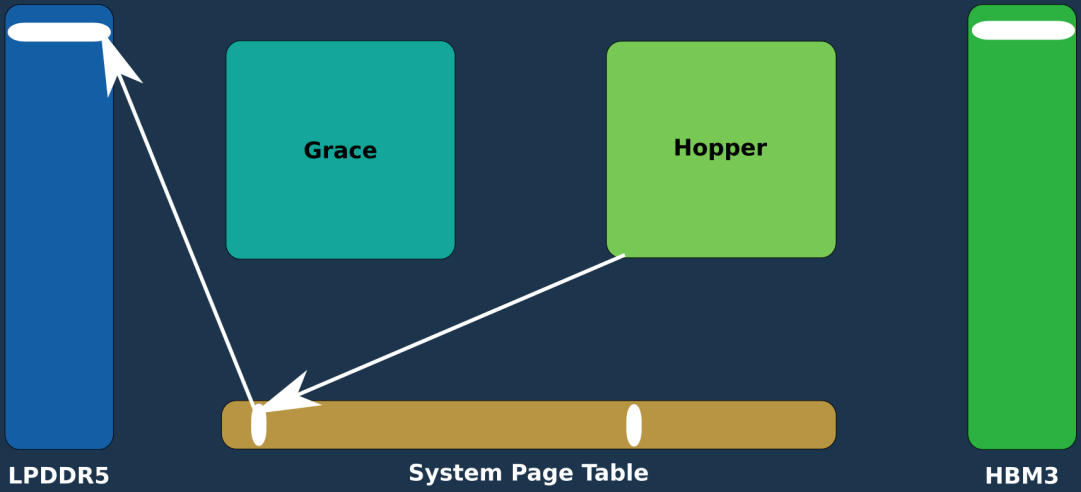
Memory Management Grace Hopper



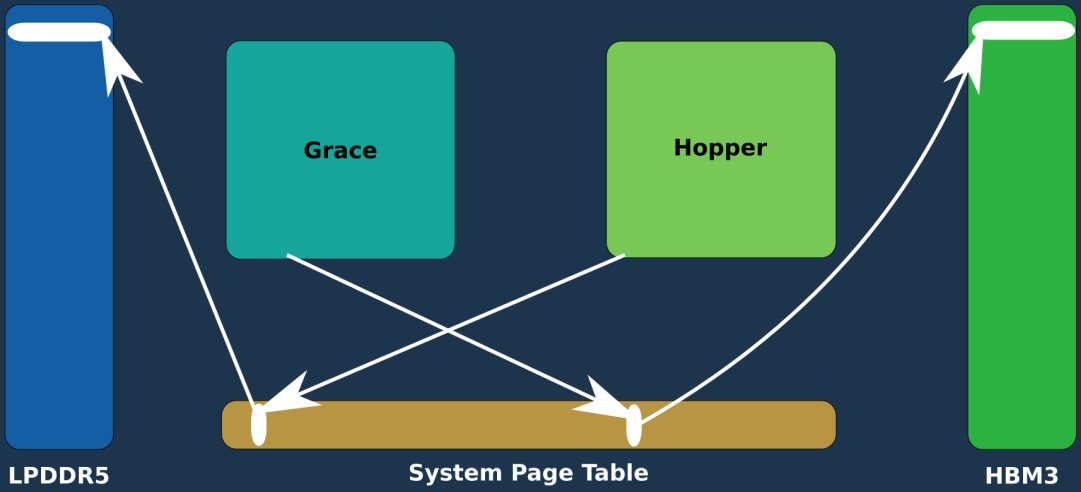
Memory Management Grace Hopper



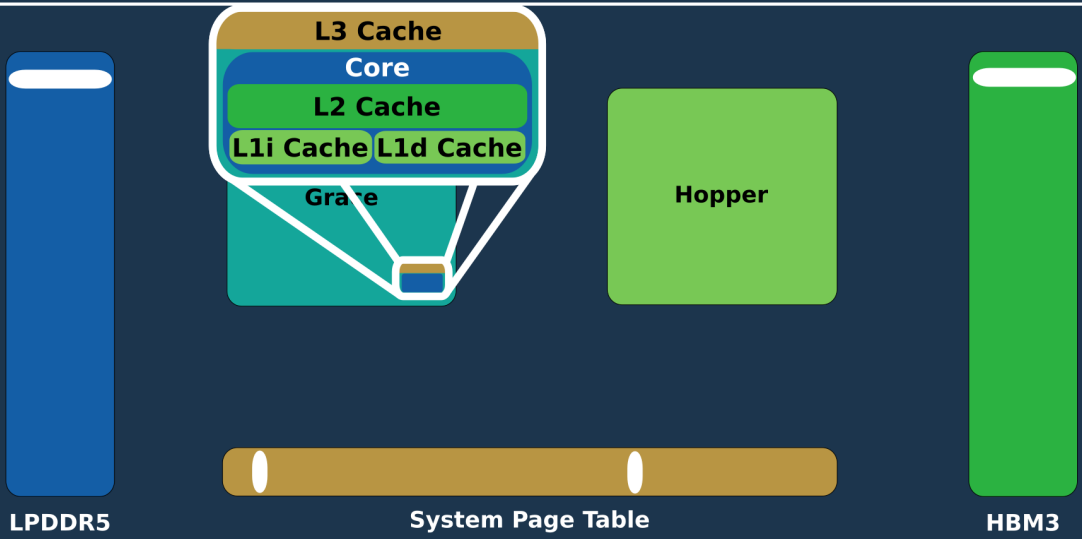
Memory Management Grace Hopper



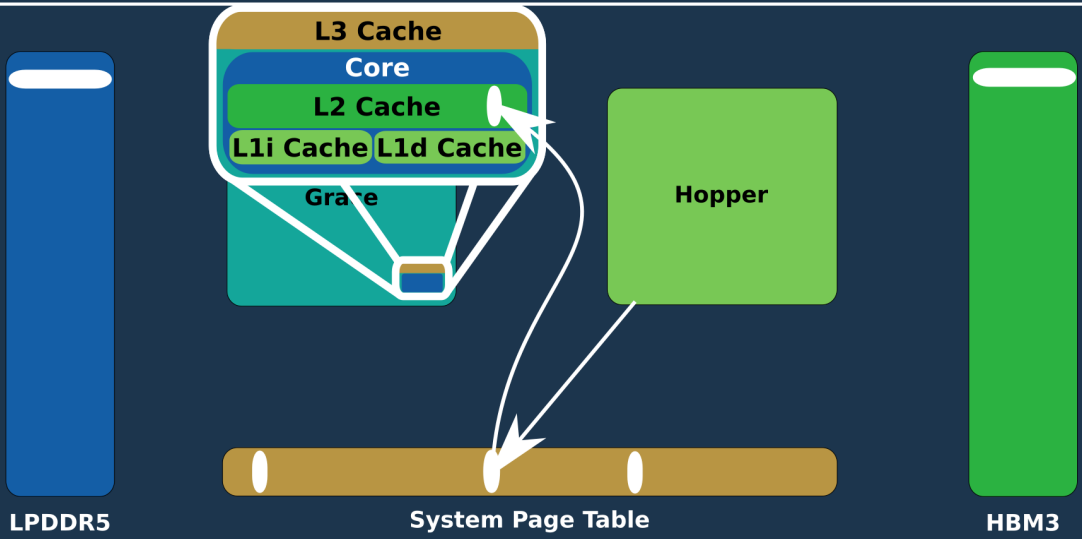
Memory Management Grace Hopper



Memory Management Grace Hopper



Memory Management Grace Hopper



Grace Software Ecosystem

Grace Software Ecosystem

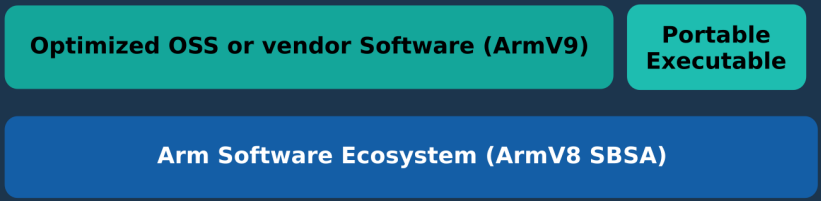
Arm Software Ecosystem (ArmV8 SBSA)

Grace Software Ecosystem

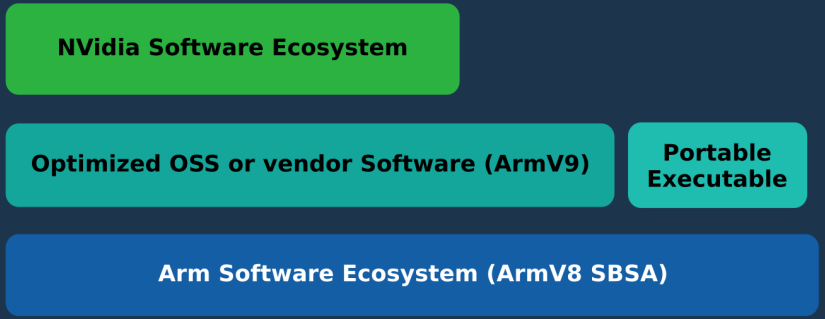
Optimized OSS or vendor Software (ArmV9)

Arm Software Ecosystem (ArmV8 SBSA)

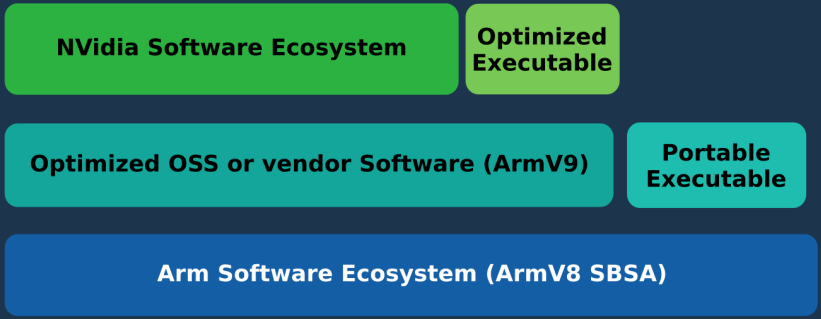
Grace Software Ecosystem



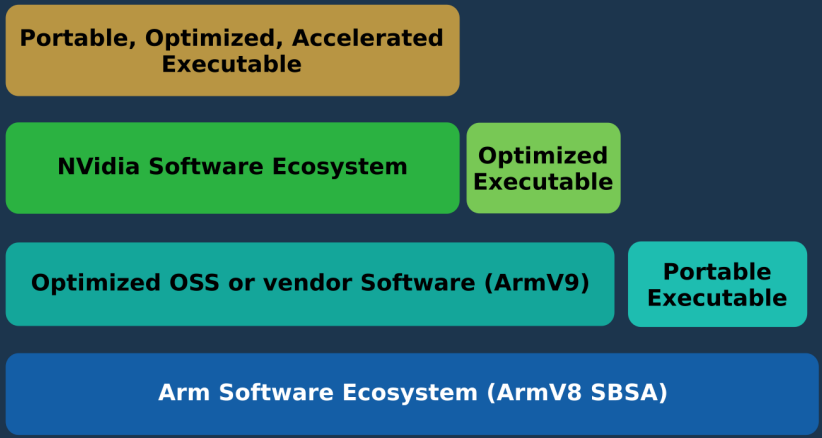
Grace Software Ecosystem



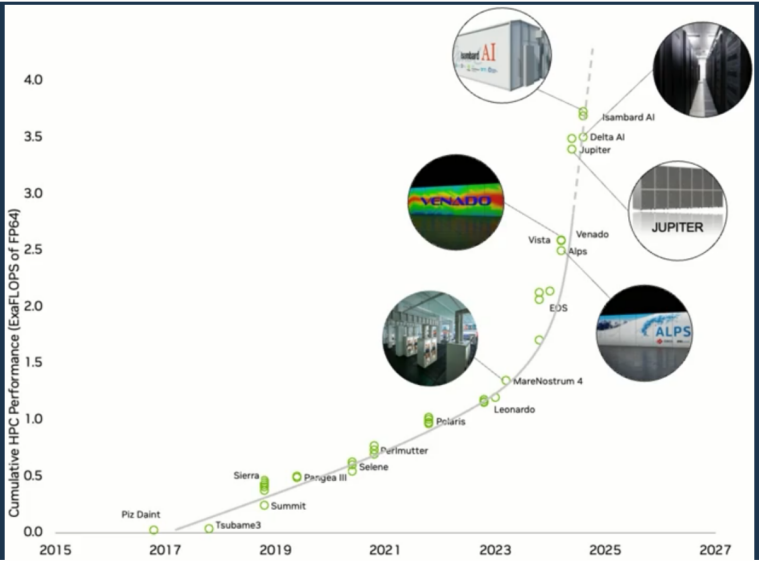
Grace Software Ecosystem



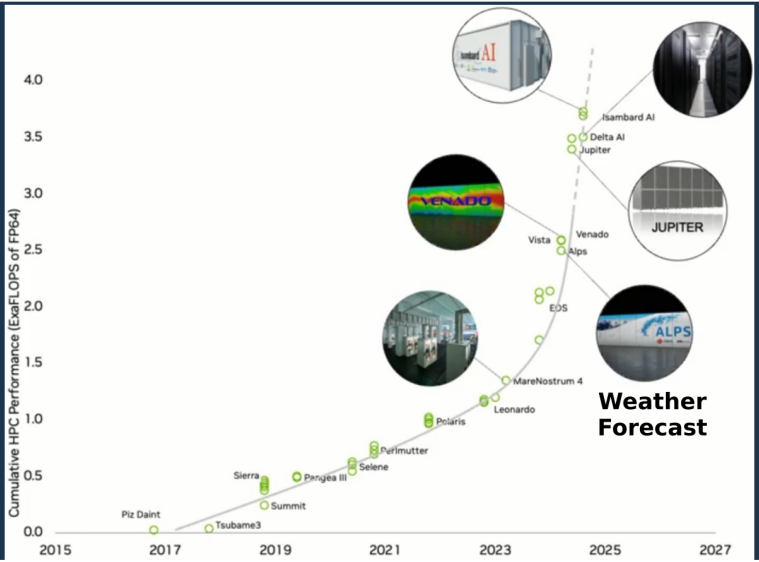
Grace Software Ecosystem



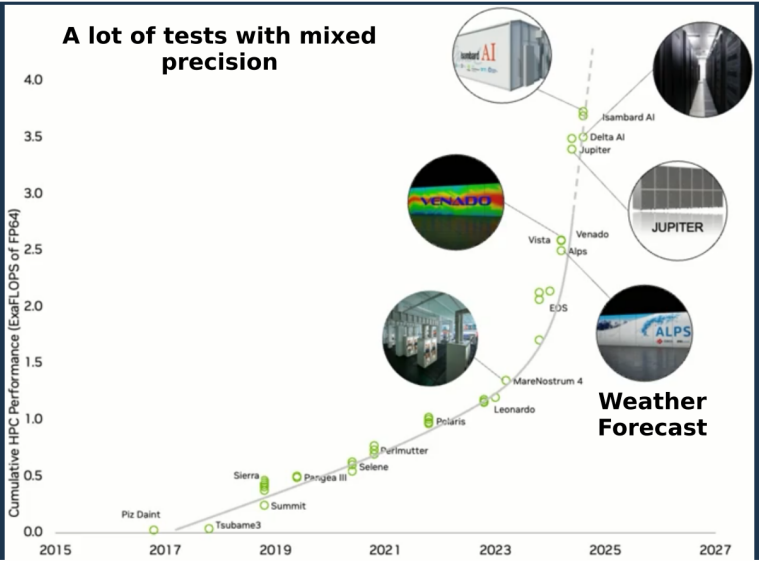
Supercomputers



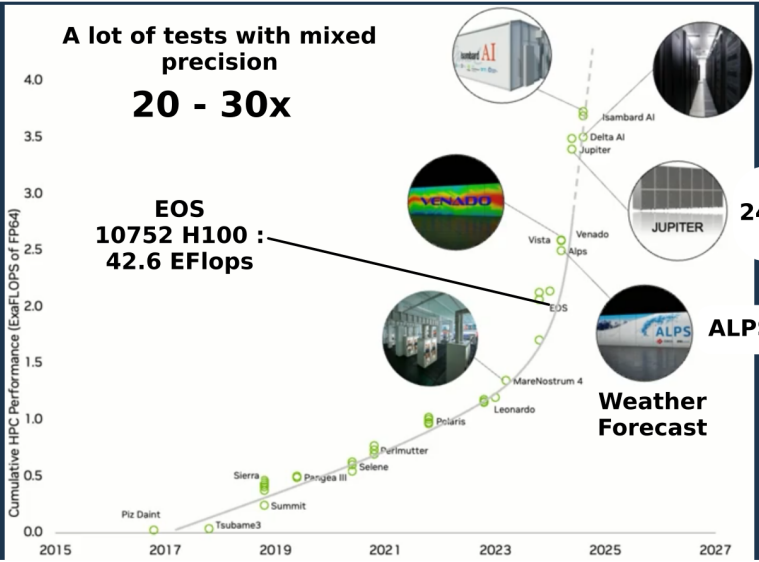
Supercomputers



Supercomputers



Supercomputers



Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

CUDA implementation (ICON)
Weather Simulation of **60 days per day**
without coupling

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

For now : 4719 PFlops (330x efficiency than Titan)

CUDA implementation (ICON)
Weather Simulation of **60 days per day without coupling**

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

For now : 4719 PFlops (330x efficiency than Titan)

CUDA implementation (ICON)
Weather Simulation of **60 days per day without coupling**

Goal :
300 days per day with Coupled Simulation
(atmosphere, ground, water, etc)

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

For now : 4719 PFlops (330x efficiency than Titan)

CUDA implementation (ICON)
Weather Simulation of **60 days per day without coupling**

Goal :
300 days per day with Coupled Simulation
(atmosphere, ground, water, etc)

With less than 5MW consuption

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

For now : 4719 PFlops (330x efficiency than Titan)

CUDA implementation (ICON)
Weather Simulation of **60 days per day without coupling**

Goal :
300 days per day with Coupled Simulation
(atmosphere, ground, water, etc)

With less than 5MW consuption

Stream tests with **1 Grace** : **375 GB/s**

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

For now : 4719 PFlops (330x efficiency than Titan)

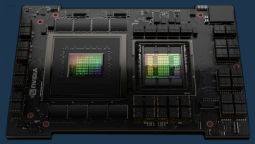
CUDA implementation (ICON)
Weather Simulation of **60 days per day without coupling**

Goal :
300 days per day with Coupled Simulation
(atmosphere, ground, water, etc)

With less than 5MW consuption

Stream tests with **1 Grace** : **375 GB/s**

Will be **archieved** :
Upgrade with
64 Grace Hopper
ready for **summer 2024**



Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

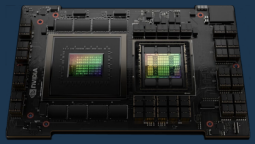
For now : 4719 PFlops (330x efficiency than Titan)

CUDA implementation (ICON)
Weather Simulation of **60 days per day without coupling**

Goal :
300 days per day with Coupled Simulation
(atmosphere, ground, water, etc)

With less than 5MW consuption

Will be **archieved** :
Upgrade with
64 Grace Hopper
ready for **summer 2024**



Stream tests with **1 Grace** : **375 GB/s**

They already investigate **mix precision** to **gain an other order of magnitude**

Feedback from ALPS

Thomas Schulthess : Director, ETH Zurich/The Swiss National Supercomputing Center (CSCS)

Forecasting weather simulation

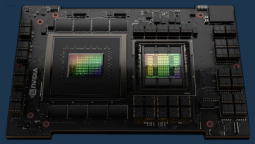
For now : 4719 PFlops (330x efficiency than Titan)

CUDA implementation (ICON)
Weather Simulation of **60 days per day without coupling**

Goal :
300 days per day with Coupled Simulation
(atmosphere, ground, water, etc)

With less than 5MW consuption

Will be **archieved** :
Upgrade with **64 Grace Hopper** ready for **summer 2024**



Stream tests with **1 Grace** : **375 GB/s**

They already investigate **mix precision** to **gain an other order of magnitude**

Next Upgrade : **40 EFlops**
2800x efficiency than **Titan**

Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray Radioastronomy (Fast radio burst, dark matter)

Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray Radioastronomy (Fast radio burst, dark matter)

LOFAR : 100s antenna => 13 Tb/s

Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray **Radioastronomy (Fast radio burst, dark matter)**

LOFAR : 100s antenna => 13 Tb/s

Group of Antenna

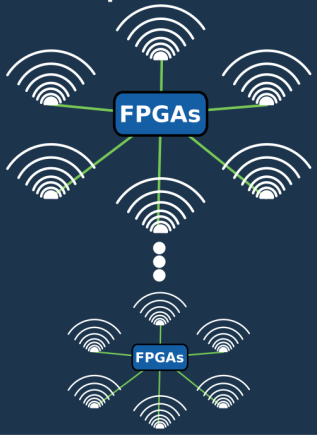
Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray **Radioastronomy (Fast radio burst, dark matter)**

LOFAR : 100s antenna => 13 Tb/s

Group of Antenna

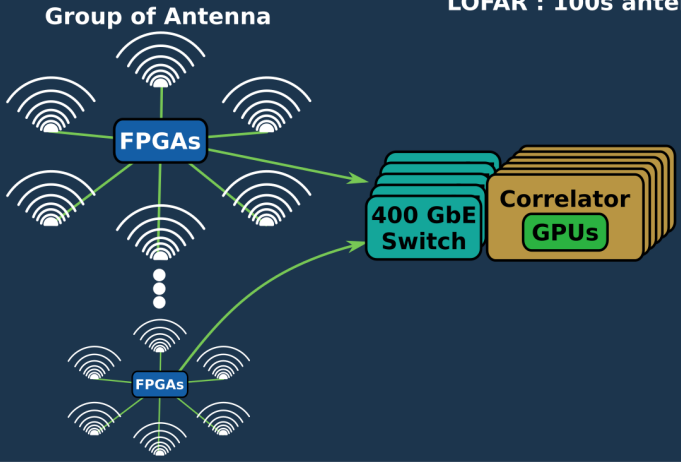


Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray **Radioastronomy (Fast radio burst, dark matter)**

LOFAR : 100s antenna => 13 Tb/s

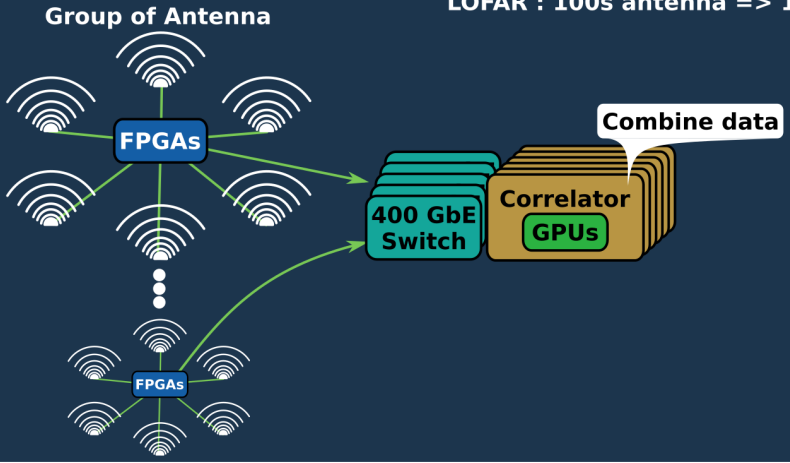


Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray **Radioastronomy (Fast radio burst, dark matter)**

LOFAR : 100s antenna => 13 Tb/s

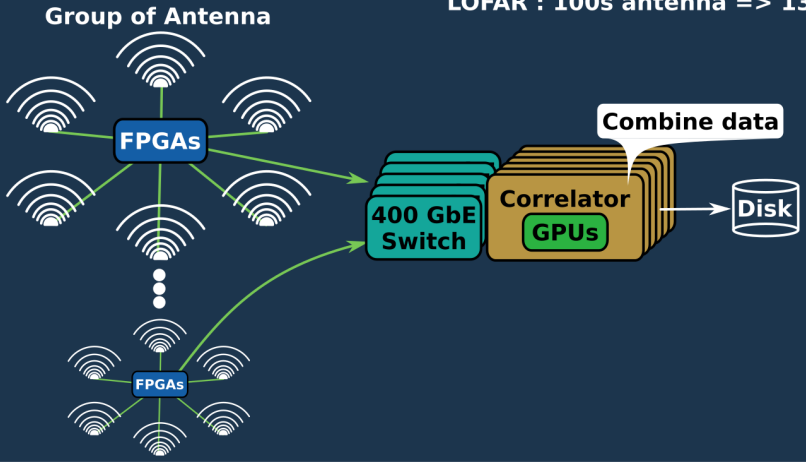


Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray **Radioastronomy (Fast radio burst, dark matter)**

LOFAR : 100s antenna => 13 Tb/s

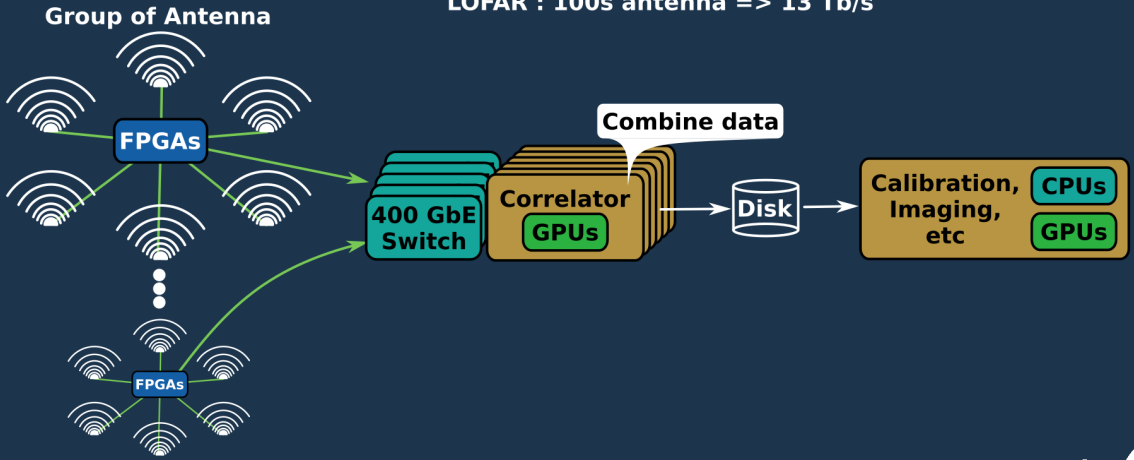


Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray Radioastronomy (Fast radio burst, dark matter)

LOFAR : 100s antenna => 13 Tb/s

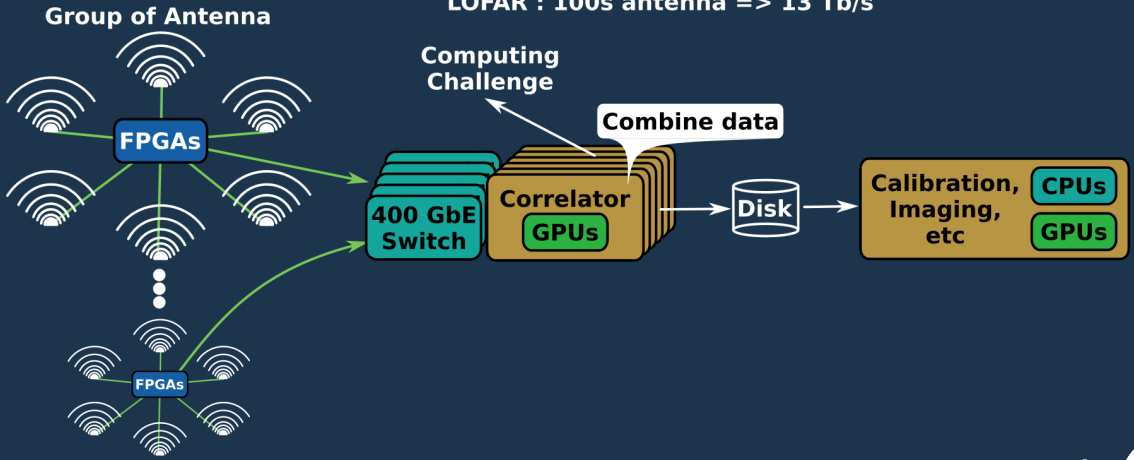


Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

LOFAR : LRw Frequency ARray **Radioastronomy (Fast radio burst, dark matter)**

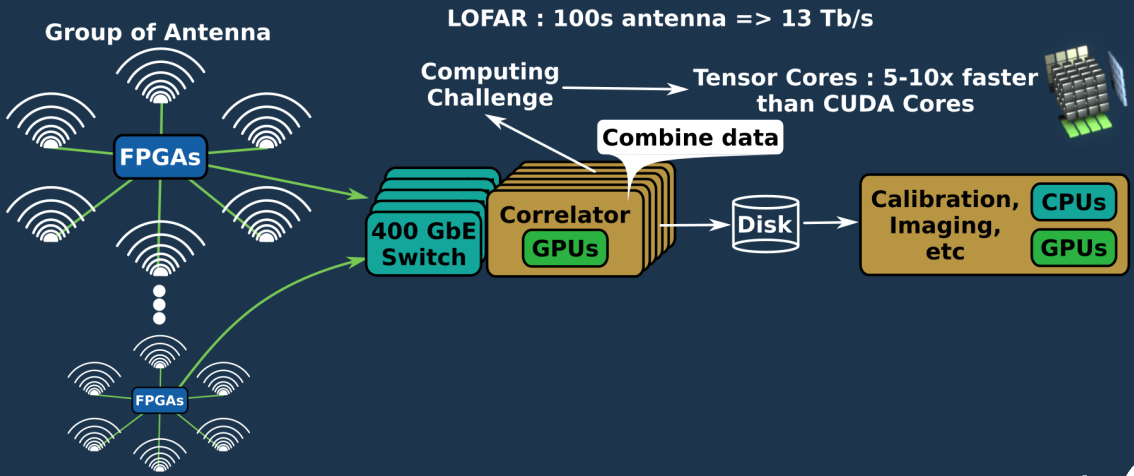
LOFAR : 100s antenna => 13 Tb/s



Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

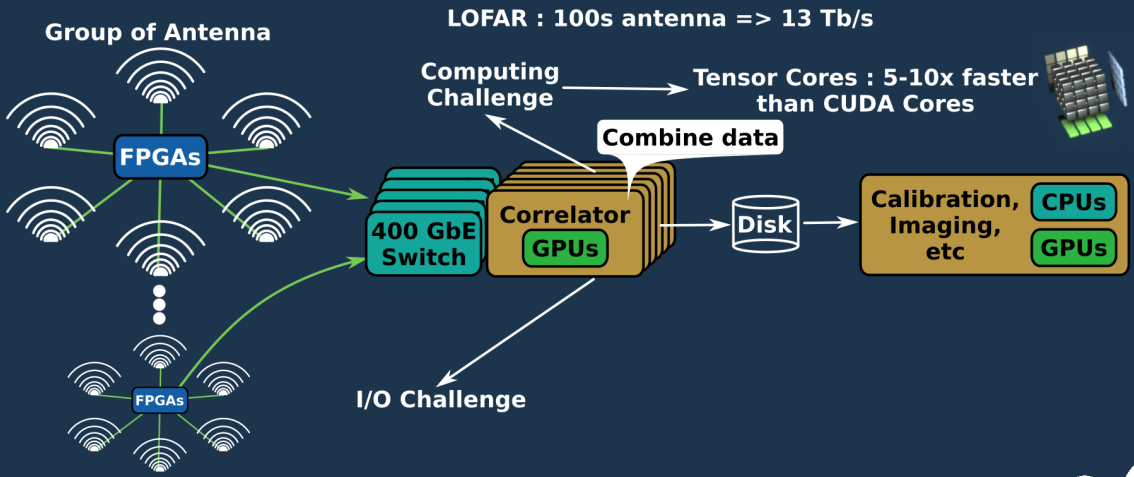
LOFAR : LRw Frequency ARray Radioastronomy (Fast radio burst, dark matter)



Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

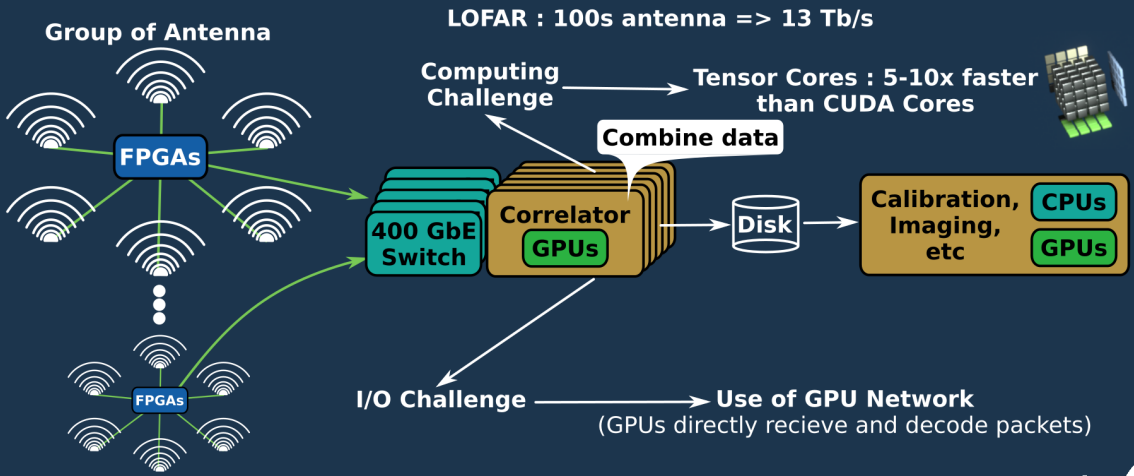
LOFAR : LRw Frequency ARray Radioastronomy (Fast radio burst, dark matter)



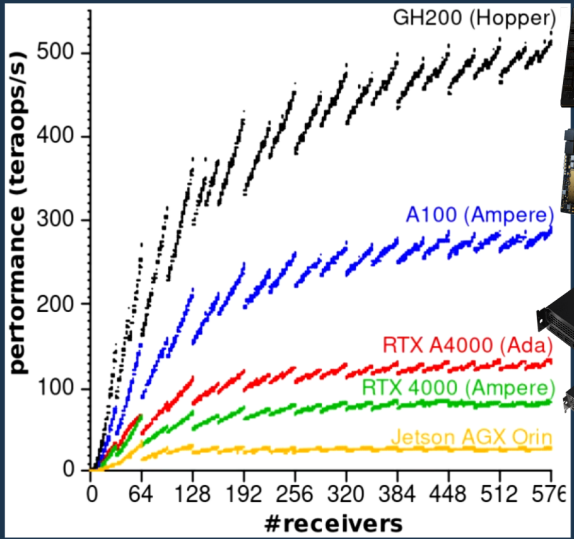
Feedback from LOFAR

John Romein : Researcher, ASTRON (Netherlands Institute for Radio Astronomy)

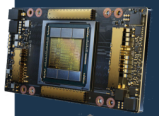
LOFAR : LRw Frequency ARray Radioastronomy (Fast radio burst, dark matter)



Feedback from LOFAR



(**TMA** not yet used)



Computing Challenge

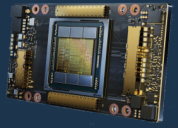
8-bit samples :
- 4-bit : ~2x faster
- 16-bit : ~2x slower

Feedback from LOFAR

I/O Challenge ⟶ Use of GPU Network

Feedback from LOFAR

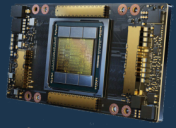
I/O Challenge → **Use of GPU Network**



A100 : (on 2 x 100Gb/s lines) get 198.6 Gb/s into A100
=> carefull tunning to avoid packet loss

Feedback from LOFAR

I/O Challenge → Use of GPU Network

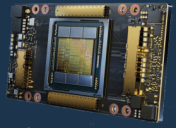


A100 : (on 2 x 100Gb/s lines) get 198.6 Gb/s into A100
=> carefull tunning to avoid packet loss

Jetson : 100 GbE NIC in PCIe slot => 99.6 Gb/s on one 100 Gb/s line with additionnal packet copy

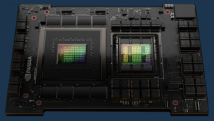
Feedback from LOFAR

I/O Challenge ⟶ Use of GPU Network



A100 : (on 2 x 100Gb/s lines) get 198.6 Gb/s into A100
=> carefull tunning to avoid packet loss

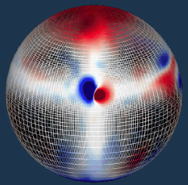
Jetson : 100 GbE NIC in PCIe slot => 99.6 Gb/s on one 100 Gb/s line with additionnal packet copy



GH200 : 398.6 Gb/s on line at 400 Gb/s
=> eventual goal 1200 Gb/s

Full Site :
13000 Gb/s

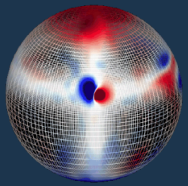
Solar Storms Simulation



Ron Caplan : Computational Scientist, Predictive Science Inc.

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.

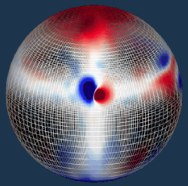


Fortran

OPEN MPI

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



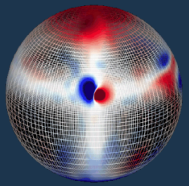
do concurrent

nvfortran

Fortran

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



do concurrent

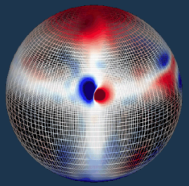
nvfortran

Fortran

Smaller code base
CPU/GPU parallelism

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



do concurrent

NVIDIA HPC DEV

nvfortran

Fortran

Smaller code base
CPU/GPU parallelism

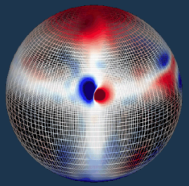
OPEN MPI

GCC

gfortran

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



do concurrent

NVIDIA HPC DEV

nvfortran

Fortran

Smaller code base
CPU/GPU parallelism

OPEN MPI

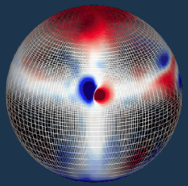
GCC

gfortran

No reduction in
do concurrent for now

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



do concurrent

NVIDIA HPC DEV

nvfortran

Fortran

Smaller code base
CPU/GPU parallelism

OPEN MPI

OpenACC
More Science, Less Programming

Data Movement
Directives

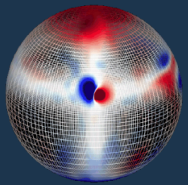
GCC

No reduction in
do concurrent for now

gfortran

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



do concurrent

nvfortran

V100

×8

34.3 h

Fortran

Smaller code base
CPU/GPU parallelism

OpenACC
More Science, Less Programming

Data Movement
Directives

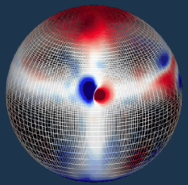
GCC

gfortran

No reduction in
do concurrent for now

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



do concurrent

nvfortran

Fortran

Smaller code base
CPU/GPU parallelism

OpenACC
More Science, Less Programming

Data Movement
Directives

GCC

gfortran

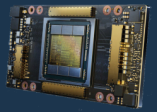
No reduction in
do concurrent for now

V100

×8

34.3 h

A100 40GB

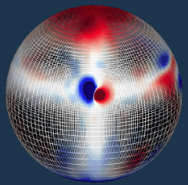


×8

23.7 h

Solar Storms Simulation

Ron Caplan : Computational Scientist, Predictive Science Inc.



do concurrent

NVIDIA HPC DEV

nvfortran

Fortran

Smaller code base
CPU/GPU parallelism

OPEN MPI

OpenACC
More Science, Less Programming

Data Movement
Directives

GCC

gfortran

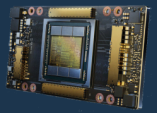
No reduction in
do concurrent for now

V100

×8

34.3 h

A100 40GB

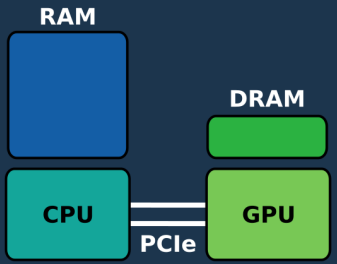


×8

23.7 h

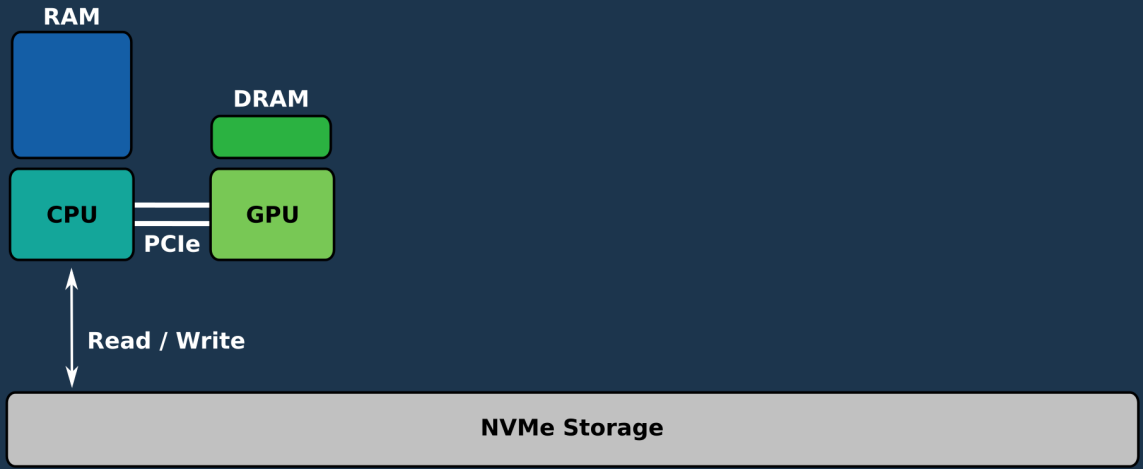
Computing needs went from
4 nodes to only
1 single node

Training with Grace Hopper



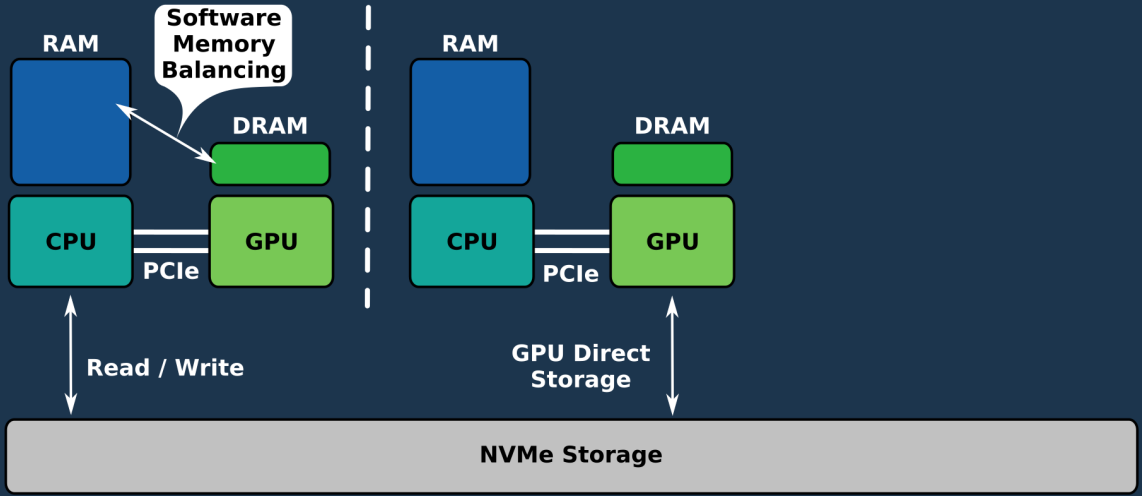
NVMe Storage

Training with Grace Hopper

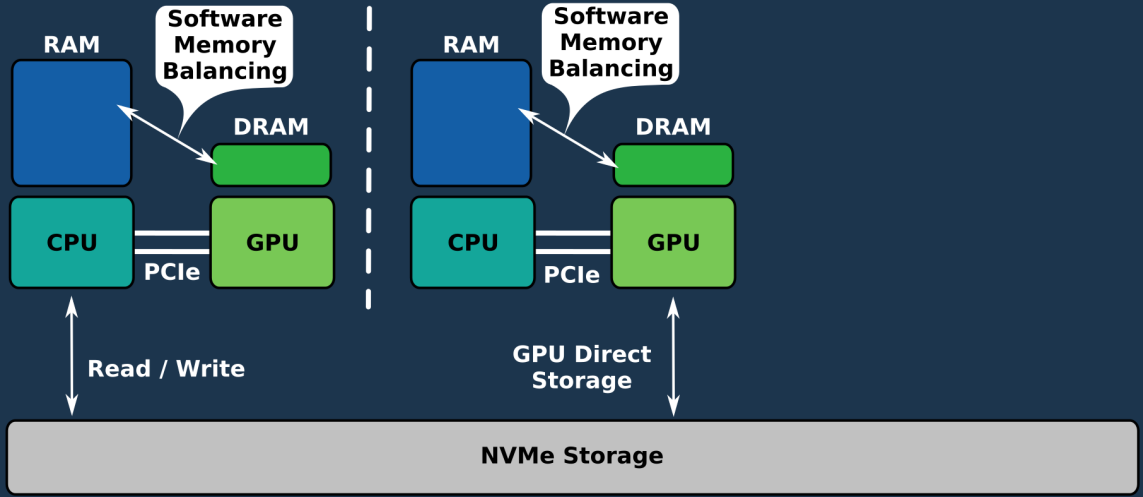


Training with Grace Hopper

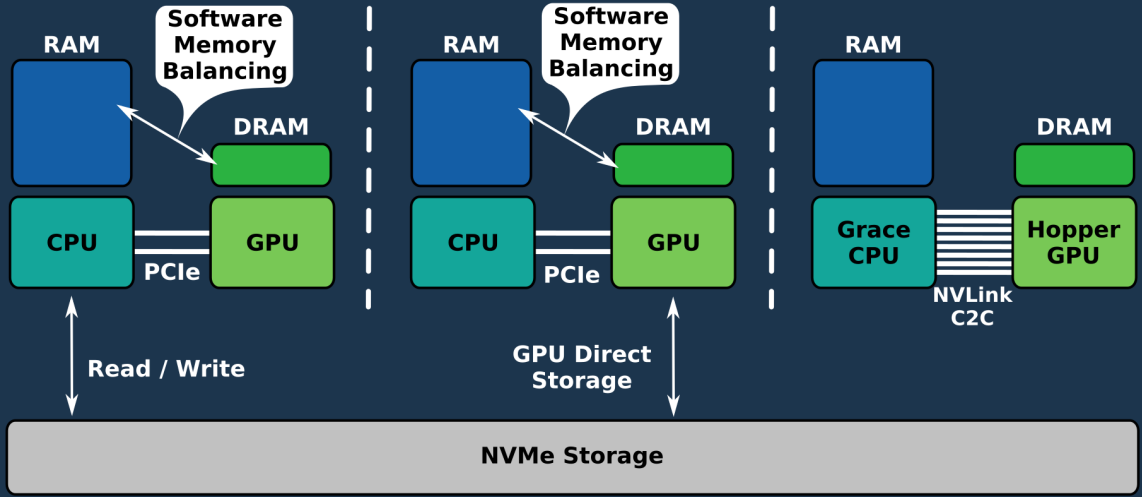
Training with Grace Hopper



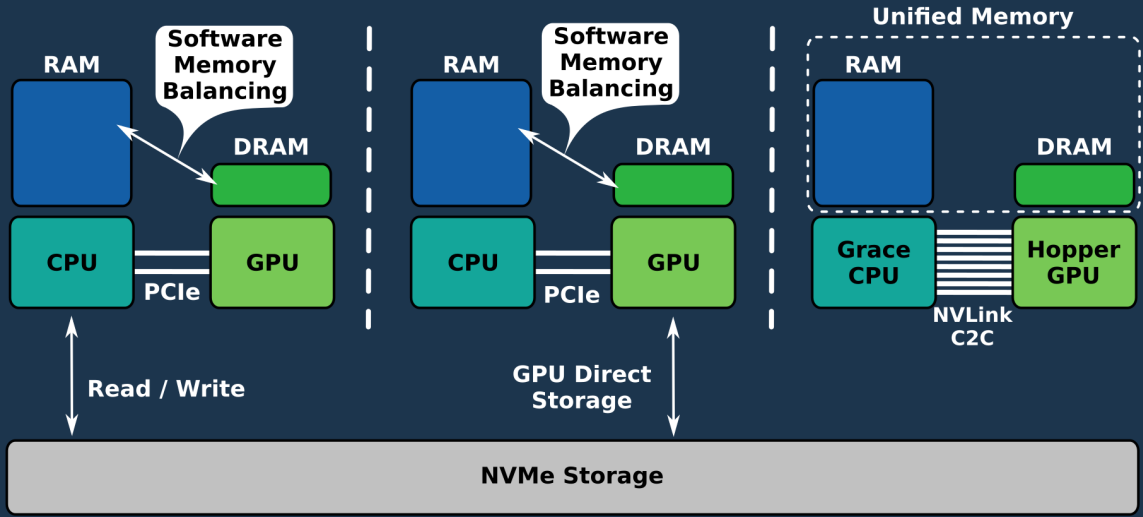
Training with Grace Hopper



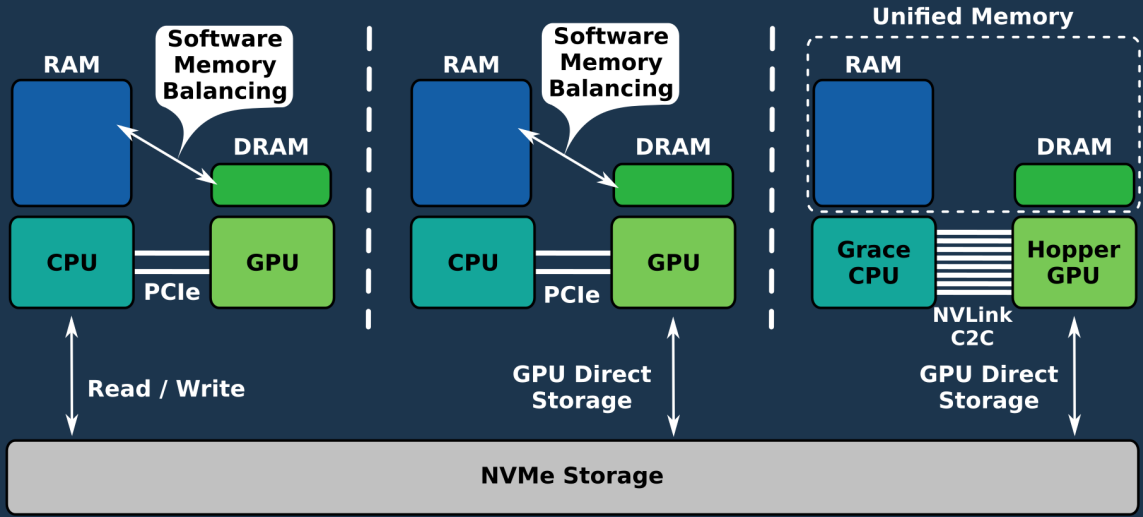
Training with Grace Hopper



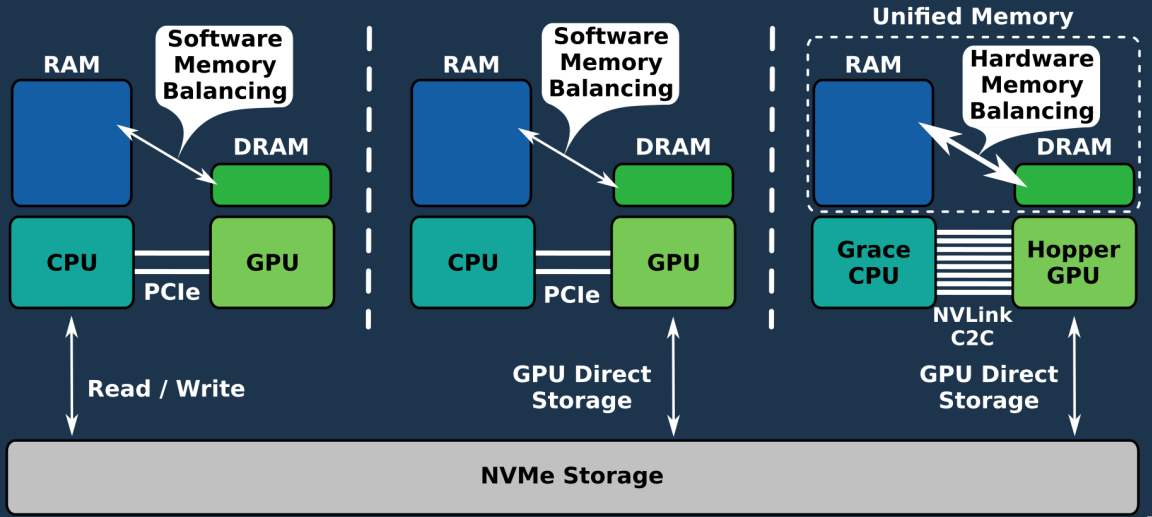
Training with Grace Hopper



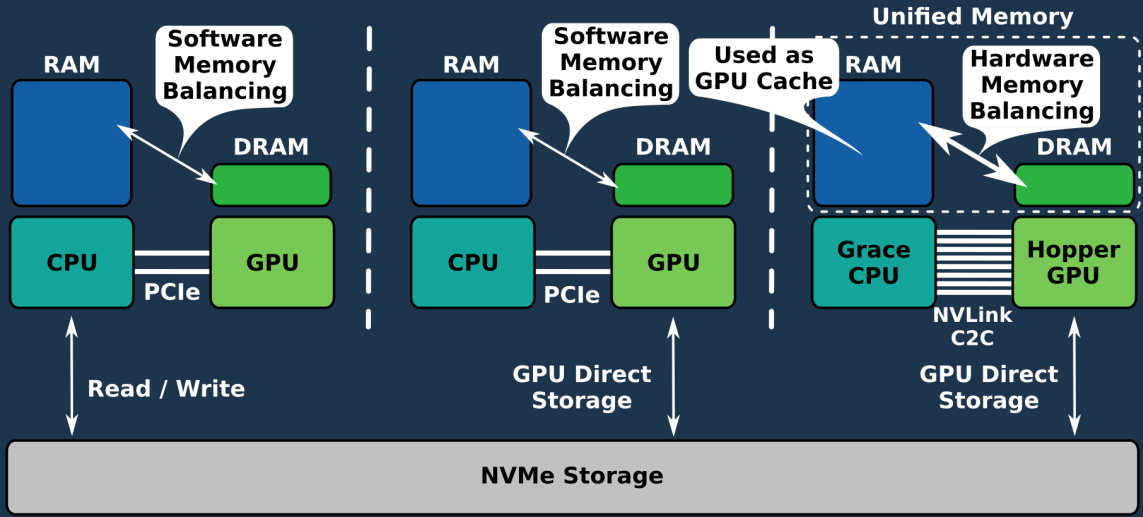
Training with Grace Hopper



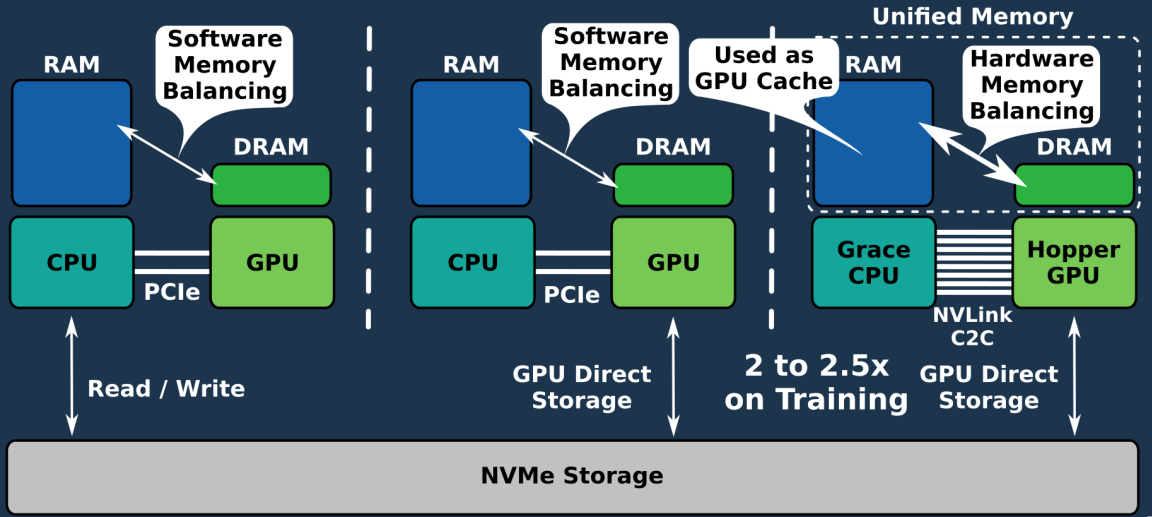
Training with Grace Hopper



Training with Grace Hopper



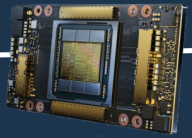
Training with Grace Hopper



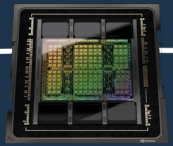
End of the first part

End of the first part

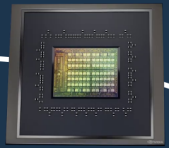
A100



H100



Grace CPU



Grace Superchip

Grace Hopper Superchip

Introduction Second Part

GTC 2024 Presentation Catalog

This is not possible to summarize **1080** talks !

Only focus on :

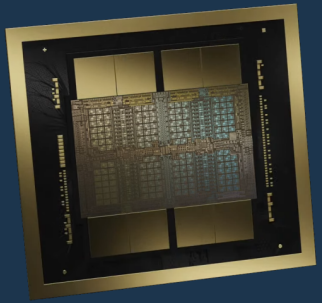
- **Hardwares** evolution
- **Frameworks** evolution :
  - **Computing**
  - **Data Formats**
  - **Machine Learning** / **Deep Learning**
- **Photolithography**

Keynotes video link

2024 - Blackwell GPU

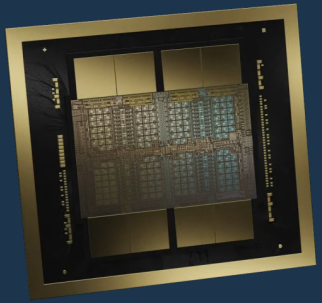
2024 - Blackwell GPU

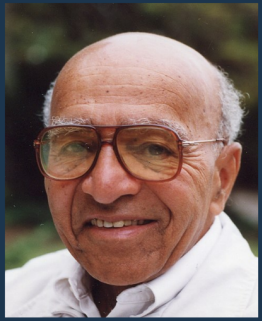
B100 (Blackwell)



2024 - Blackwell GPU

B100 (Blackwell)





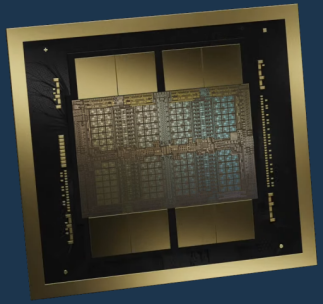
David Harold Blackwell (1919 - 2010)

Statistician and **mathematician** :
- **Game theory**
- **Probability theory**
- **Information theory**
- **Statistics**

2024 - Blackwell GPU

4 nm

B100 (Blackwell)

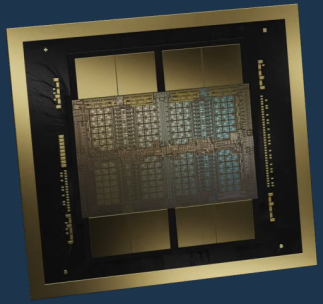


2024 - Blackwell GPU

4 nm

Compute Capabilites **10.0**

B100 (Blackwell)



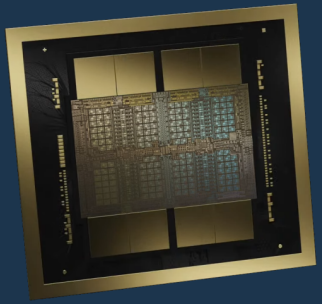
2024 - Blackwell GPU

4 nm

Compute Capabilites **10.0**

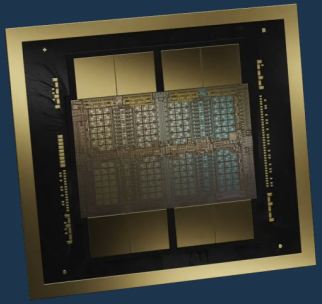
160 SMs (tensor core Gen4) 10240 Cores

B100 (Blackwell)



2024 - Blackwell GPU

B100 (Blackwell)



4 nm

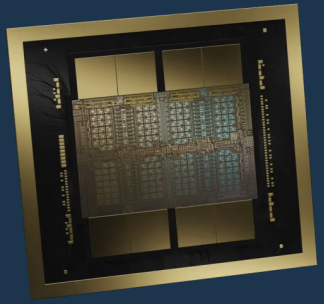
Compute Capabilites **10.0**

160 SMs (tensor core Gen4) 10240 Cores

192 GB HBM3e

2024 - Blackwell GPU

B100 (Blackwell)



4 nm

Compute Capabilites **10.0**

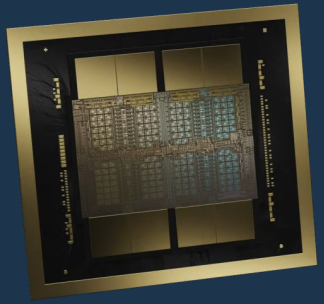
160 SMs (tensor core Gen4) 10240 Cores

192 GB HBM3e

8 TB/s Bandwidth

2024 - Blackwell GPU

B100 (Blackwell)



4 nm

Compute Capabilites **10.0**

160 SMs (tensor core Gen4) 10240 Cores

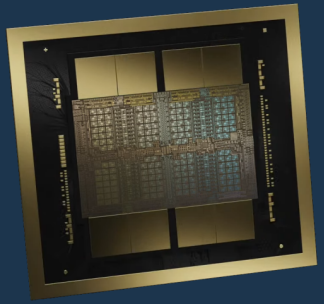
192 GB HBM3e

8 TB/s Bandwidth

NVLink Gen 5 (scale up to **576** GPUs)

2024 - Blackwell GPU

B100 (Blackwell)



4 nm

Compute Capabilites **10.0**

160 SMs (tensor core Gen4) 10240 Cores

192 GB HBM3e

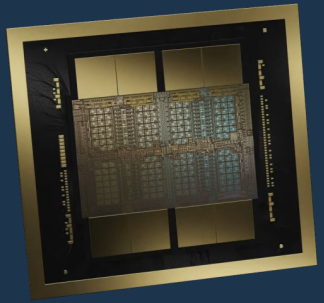
8 TB/s Bandwidth

NVLink Gen 5 (scale up to **576** GPUs)

More computing precisions

2024 - Blackwell GPU

B100 (Blackwell)



4 nm

Compute Capabilites **10.0**

160 SMs (tensor core Gen4) 10240 Cores

192 GB HBM3e

8 TB/s Bandwidth

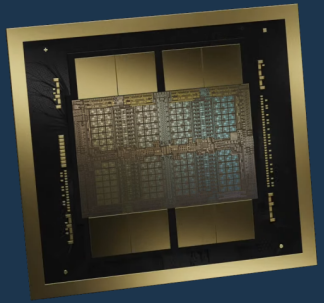
NVLink Gen 5 (scale up to **576** GPUs)

More computing precisions

Transformer Engine V2 (**FP4**/**FP6** on inference)

2024 - Blackwell GPU

B100 (Blackwell)



4 nm

Compute Capabilites **10.0**

160 SMs (tensor core Gen4) 10240 Cores

192 GB HBM3e

8 TB/s Bandwidth

NVLink Gen 5 (scale up to **576** GPUs)

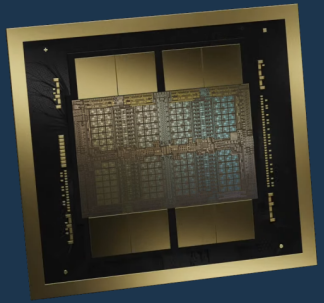
More computing precisions

Transformer Engine V2 (**FP4**/**FP6** on inference)

Decompression Engine : **800** GB/s

2024 - Blackwell GPU

B100 (Blackwell)



4 nm

Compute Capabilites **10.0**

160 SMs (tensor core Gen4) 10240 Cores

192 GB HBM3e

8 TB/s Bandwidth

NVLink Gen 5 (scale up to **576** GPUs)

More computing precisions

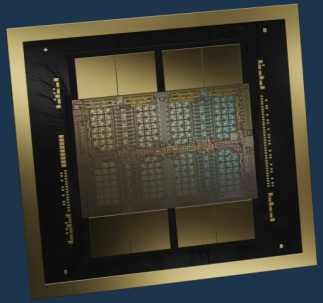
Transformer Engine V2 (**FP4**/**FP6** on inference)

Decompression Engine : **800** GB/s

Secure AI : full performance encryption

2024 - Blackwell GPU

B100 (Blackwell)



4 nm

Compute Capabilites **10.0**

160 SMs (tensor core Gen4) 10240 Cores

192 GB HBM3e

8 TB/s Bandwidth

NVLink Gen 5 (scale up to **576** GPUs)

More computing precisions

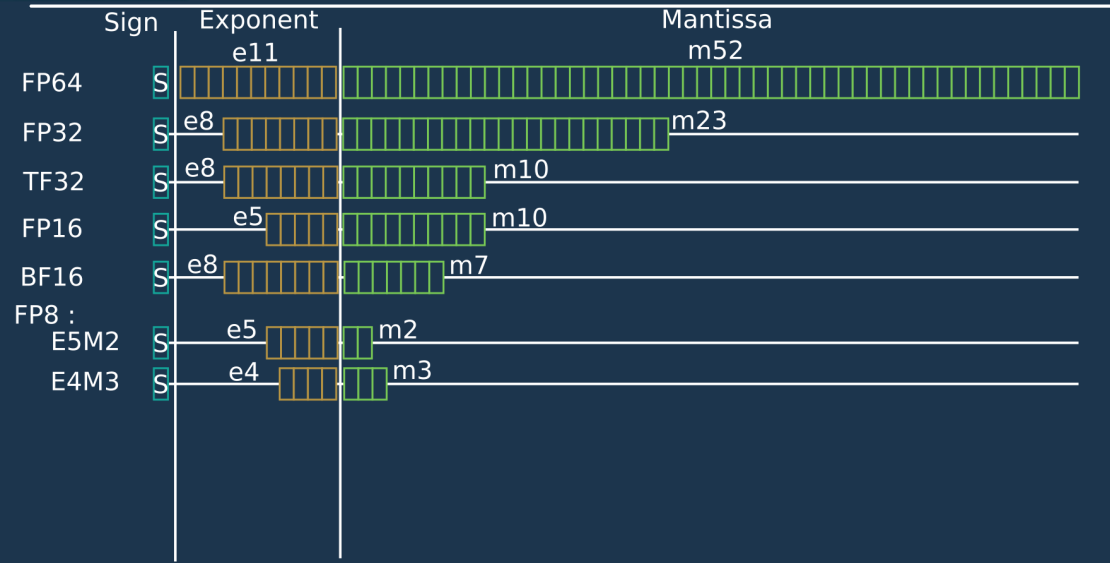
Transformer Engine V2 (**FP4**/**FP6** on inference)

Decompression Engine : **800** GB/s

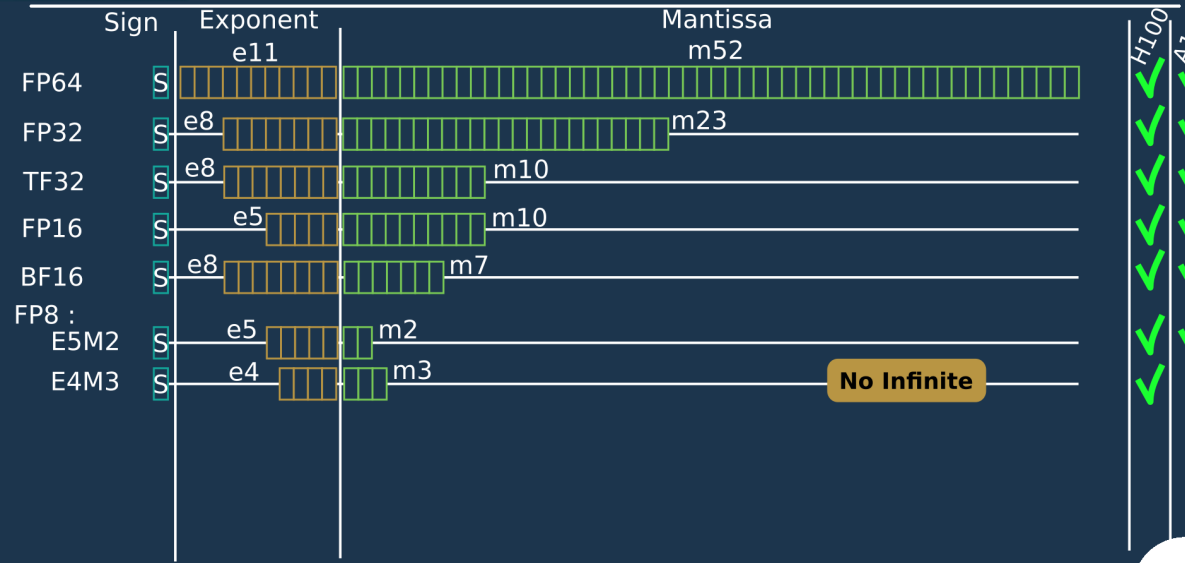
Secure AI : full performance encryption

RAS Engine

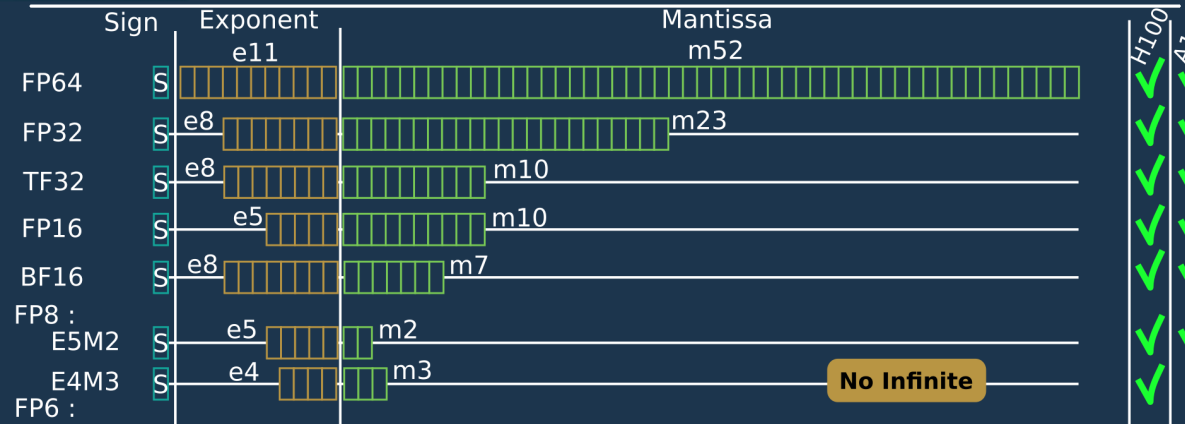
More Computing Precision



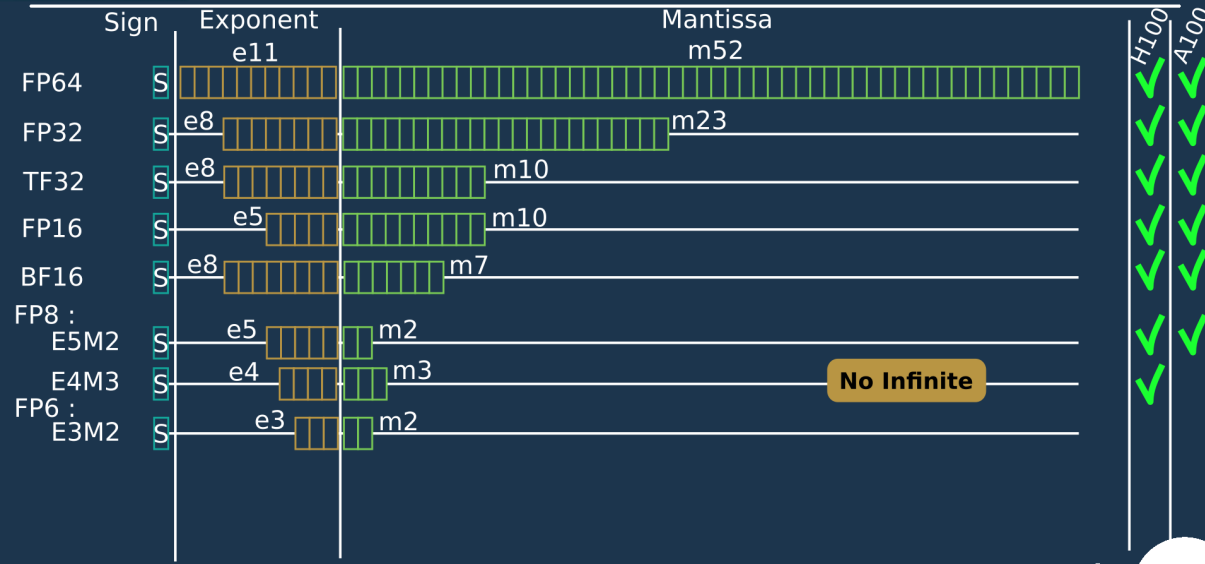
More Computing Precision



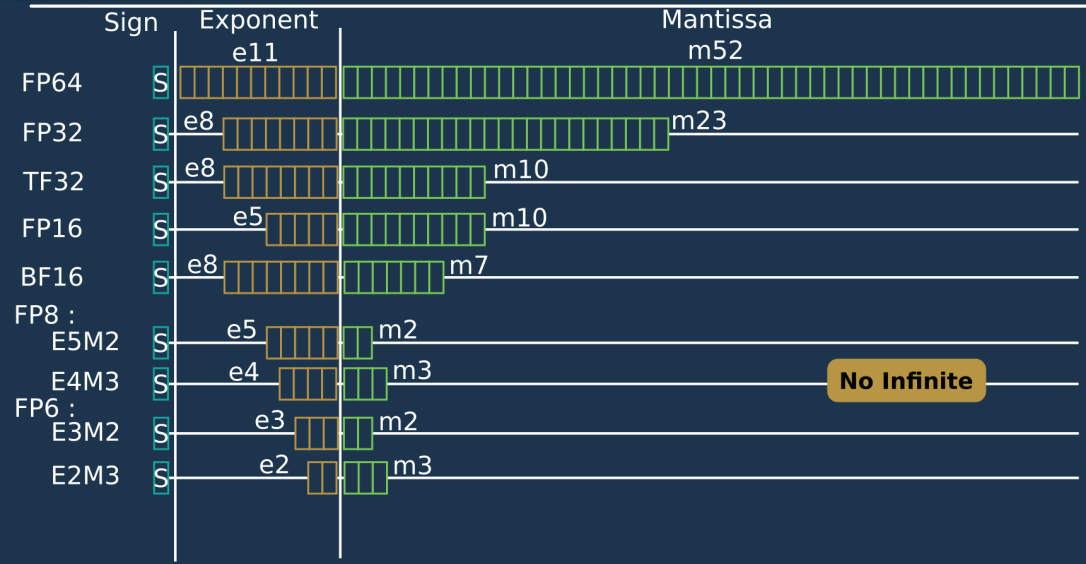
More Computing Precision



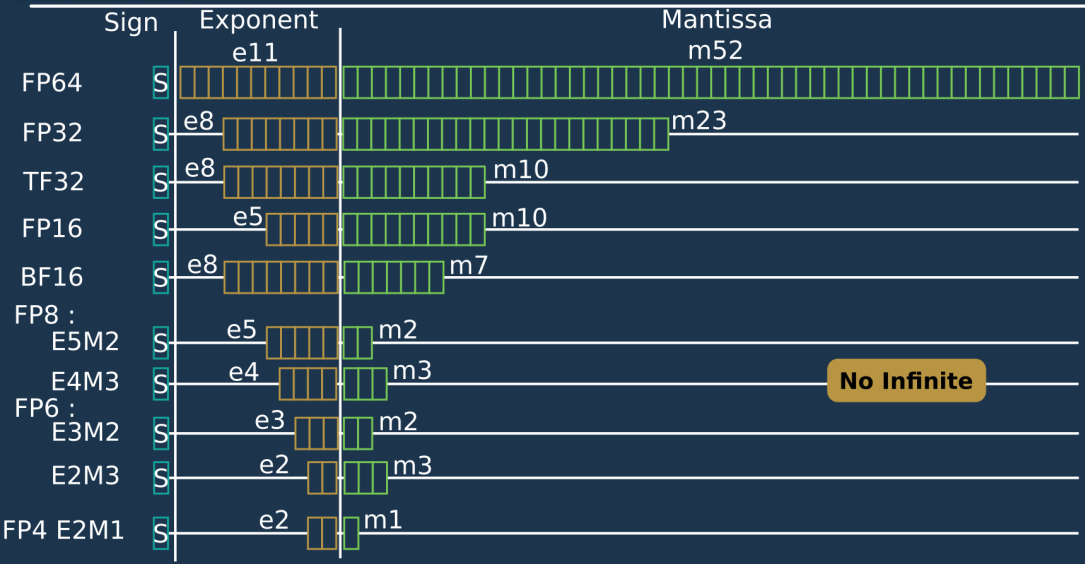
More Computing Precision



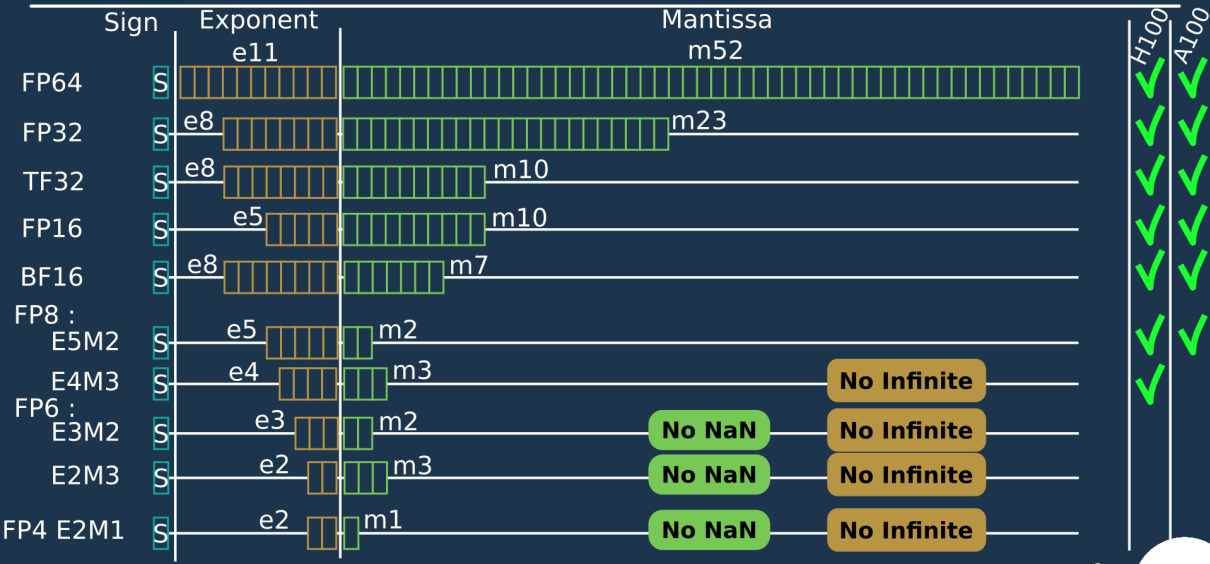
More Computing Precision



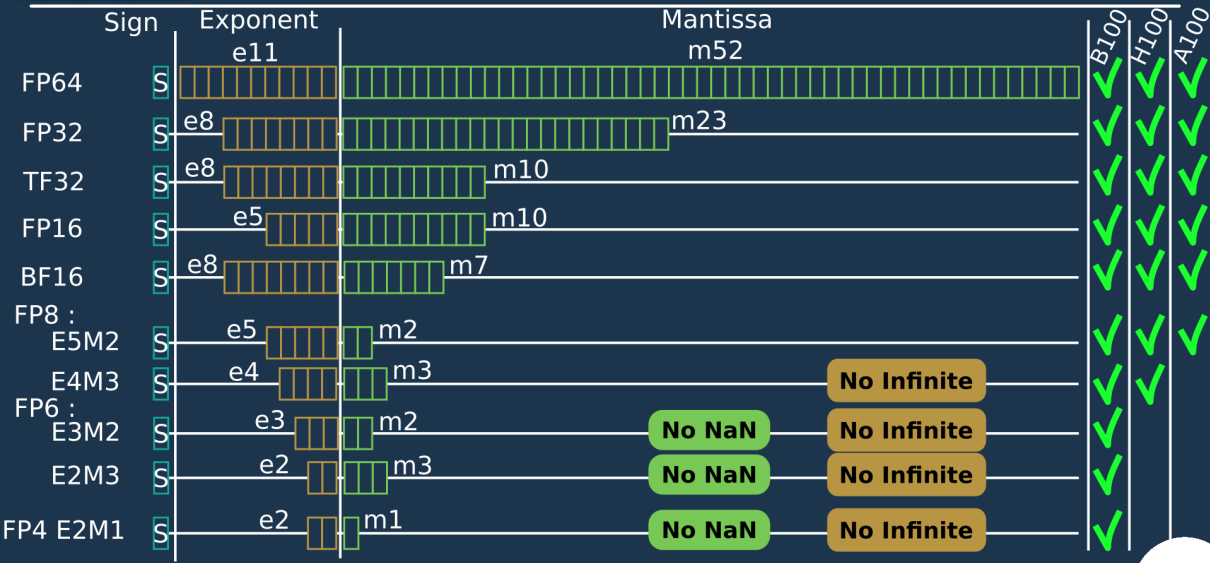
More Computing Precision



More Computing Precision



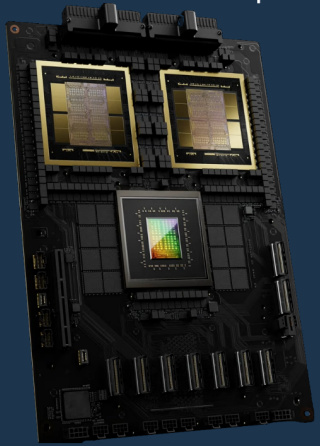
More Computing Precision



Grace Blackwell Superchip

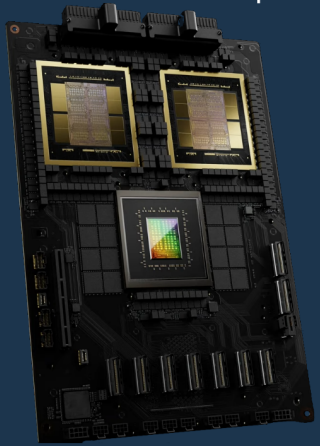
1 Grace CPU + **2 Blackwell** GPU

GB200
(Grace Blackwell Superchip)



Grace Blackwell Superchip

GB200
(Grace Blackwell Superchip)



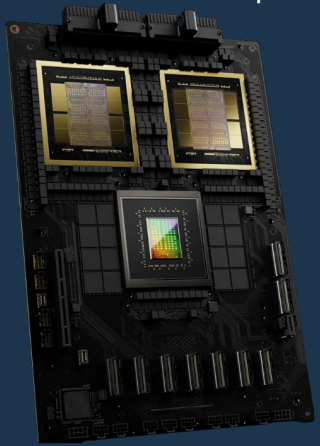
1 Grace CPU + **2 Blackwell** GPU

GPU

- **90** TFlops : **FP64** Tensor Core
- **5** PFlops : **TF32** Tensor Core
- **10** PFlops : **FP16** / **BF16** Tensor Core
- **20** PFlops : **FP8** / **FP6** Tensor Core
- **20** POps : **INT8** Tensor Core
- **40** PFlops : **FP4** Tensor Core
- **384** GB HBM3e
- **16** TB/s Bandwidth

Grace Blackwell Superchip

GB200
(Grace Blackwell Superchip)



1 Grace CPU + **2 Blackwell** GPU

GPU
- **90** TFlops : **FP64** Tensor Core
- **5** PFlops : **TF32** Tensor Core
- **10** PFlops : **FP16** / **BF16** Tensor Core
- **20** PFlops : **FP8** / **FP6** Tensor Core
- **20** POps : **INT8** Tensor Core
- **40** PFlops : **FP4** Tensor Core
- **384** GB HBM3e
- **16** TB/s Bandwidth

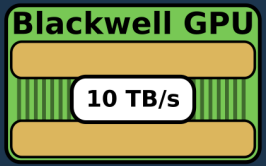
CPU (Grace **7** TFlops)
- **480** GB LPDDR5X
- **512** GB/s Bandwidth
- NVLink (3.6 TB/s)

Grace Blackwell Superchip

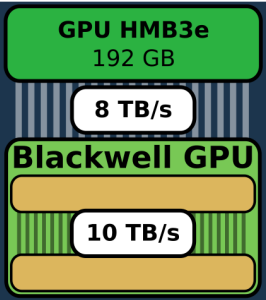
Grace Blackwell Superchip

Grace Blackwell Superchip

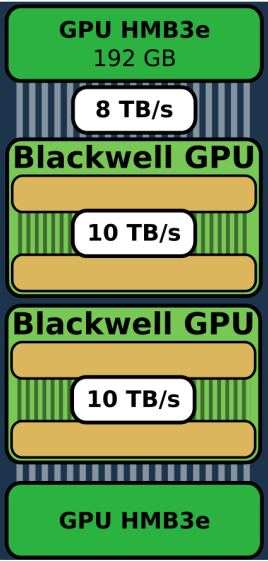
Grace Blackwell Superchip



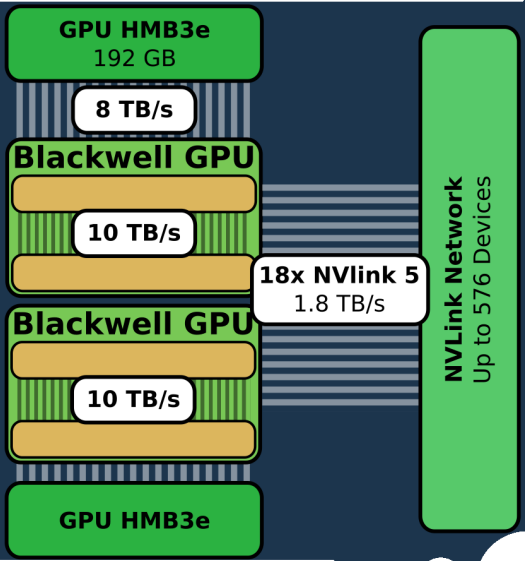
Grace Blackwell Superchip



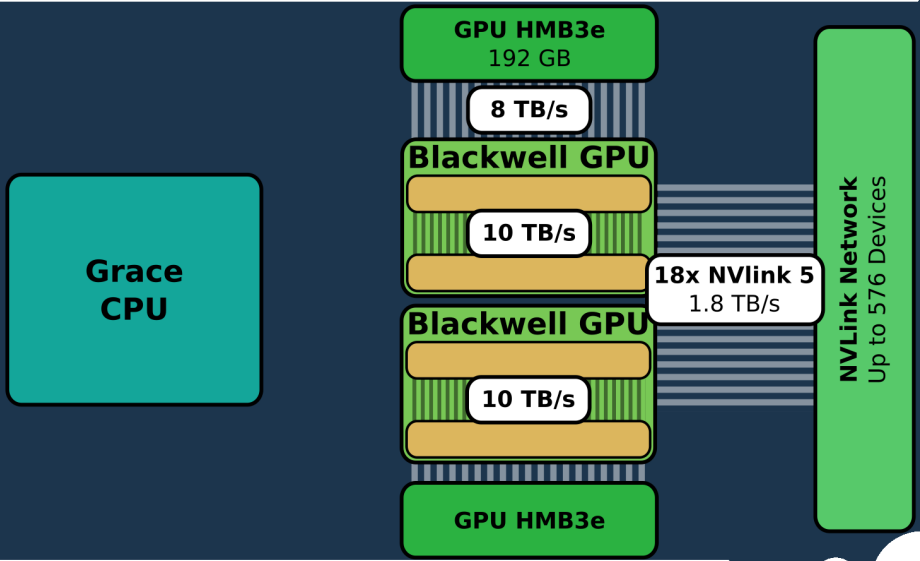
Grace Blackwell Superchip



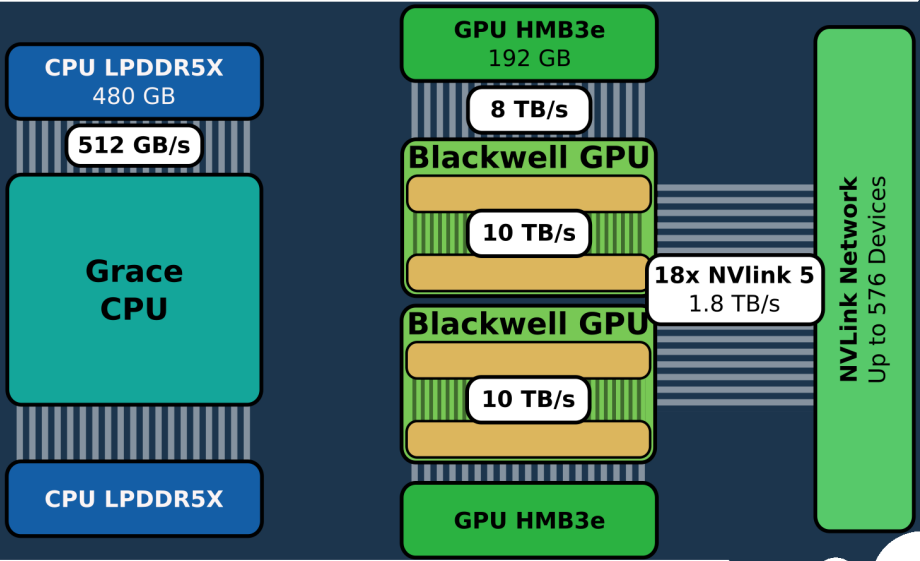
Grace Blackwell Superchip



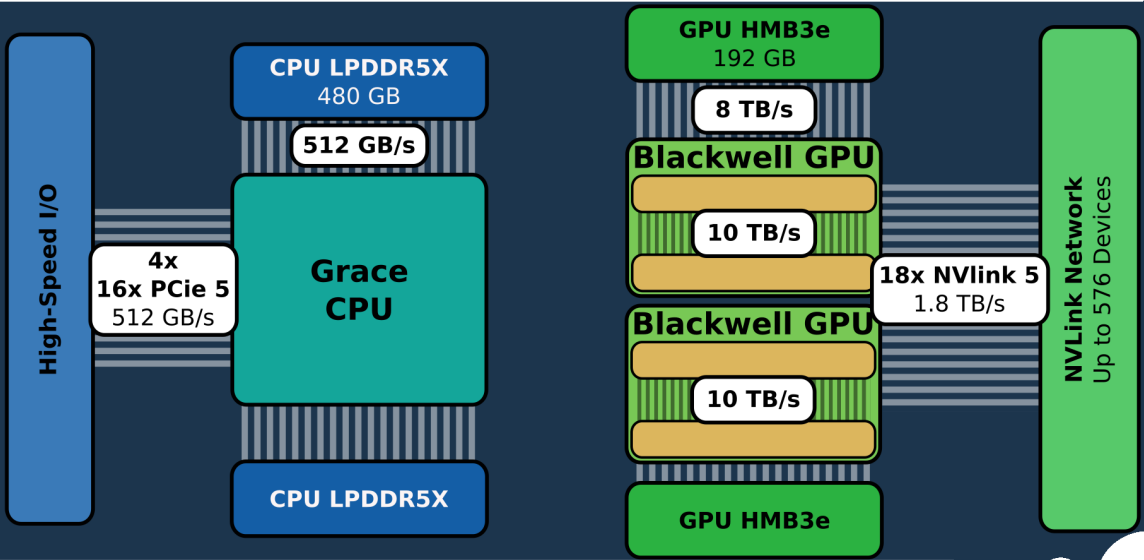
Grace Blackwell Superchip



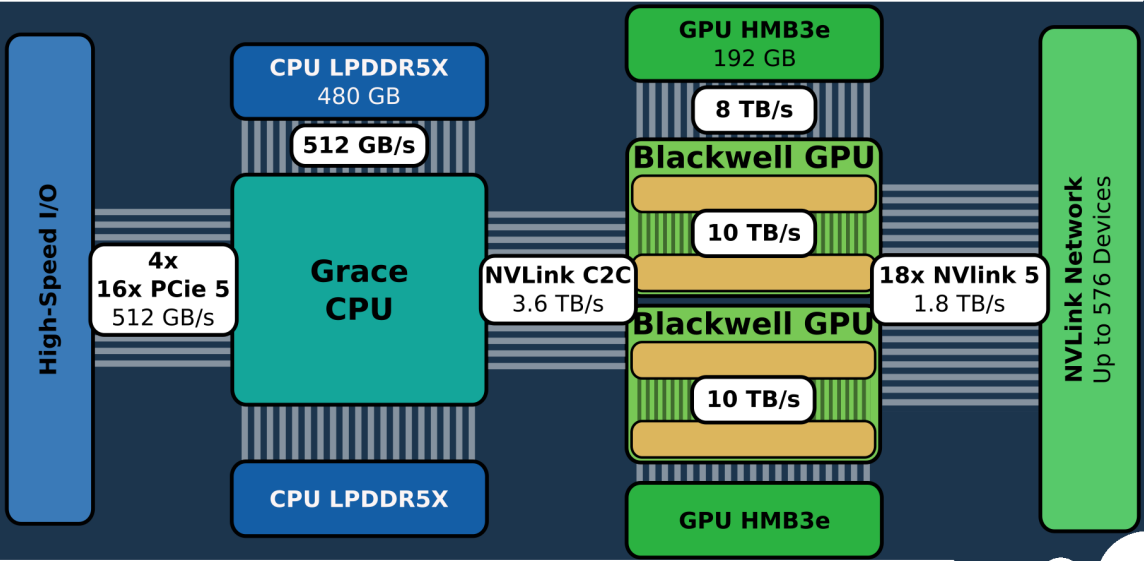
Grace Blackwell Superchip



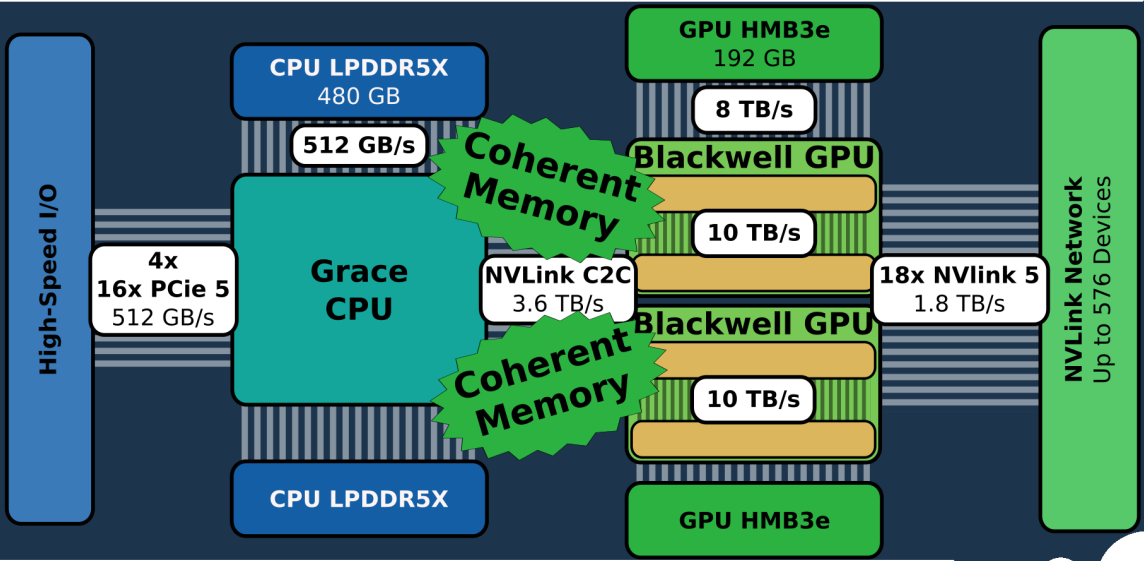
Grace Blackwell Superchip



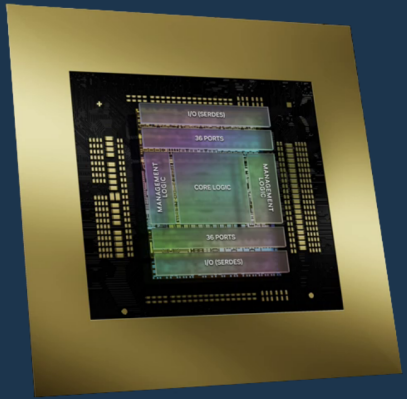
Grace Blackwell Superchip



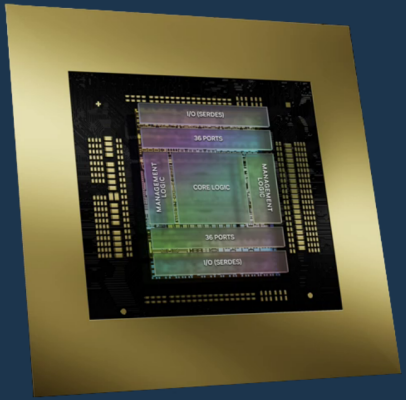
Grace Blackwell Superchip



NVLink Switch Chip

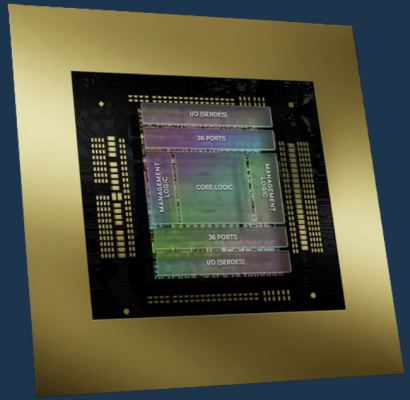


NVLink Switch Chip



50B Transistors

NVLink Switch Chip



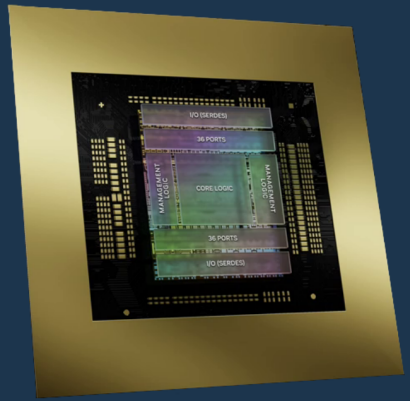
50B Transistors

72-Ports dual **200** Gb/s SerDes

4 NVlink at **1.8** TB/s

7.2 TB/s full duplex bandwidth

NVLink Switch Chip



50B Transistors

72-Ports dual **200** Gb/s SerDes

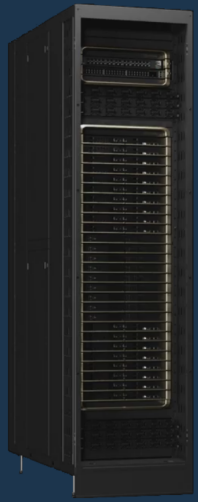
4 NVlink at **1.8** TB/s

7.2 TB/s full duplex bandwidth

Sharp-In Network compute **3.6** TFlops **FP8**

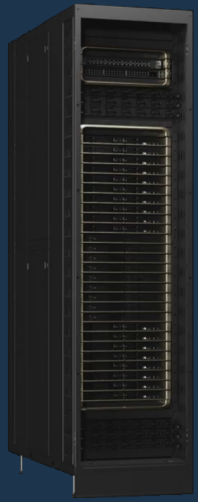
Blackwell Rack

Blackwell Rack



Blackwell Rack

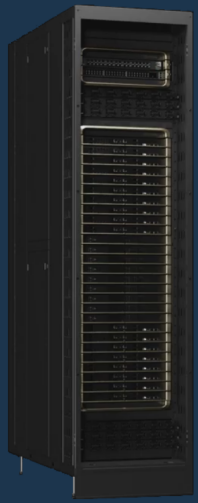
Blackwell Rack



36 Grace CPU + **72 Blackwell** GPU

Blackwell Rack

Blackwell Rack



36 Grace CPU + **72 Blackwell** GPU

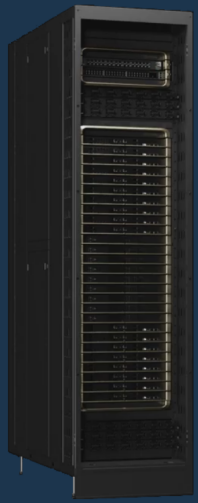
Mixed Precision Performance :

720 PFlops : Training

1440 PFlops : Inference (with **FP4**/**FP6** and sparsity)

Blackwell Rack

Blackwell Rack



36 Grace CPU + **72 Blackwell** GPU

Mixed Precision Performance :

720 PFlops : Training

1440 PFlops : Inference (with **FP4**/**FP6** and sparsity)

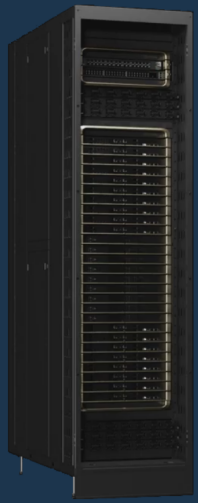
Water Cooling : in **25**°C, out **45**°C at **2** l/s

Weight : **1.5** t (**600 000** parts)

5000 NVLink cables

Blackwell Rack

Blackwell Rack



36 Grace CPU + **72 Blackwell** GPU

Mixed Precision Performance :

720 PFlops : Training

1440 PFlops : Inference (with **FP4**/**FP6** and sparsity)

Water Cooling : in **25**°C, out **45**°C at **2** l/s

Weight : **1.5** t (**600 000** parts)

5000 NVLink cables

First EFlop Rack

Blackwell Plateform

Blackwell Plateform

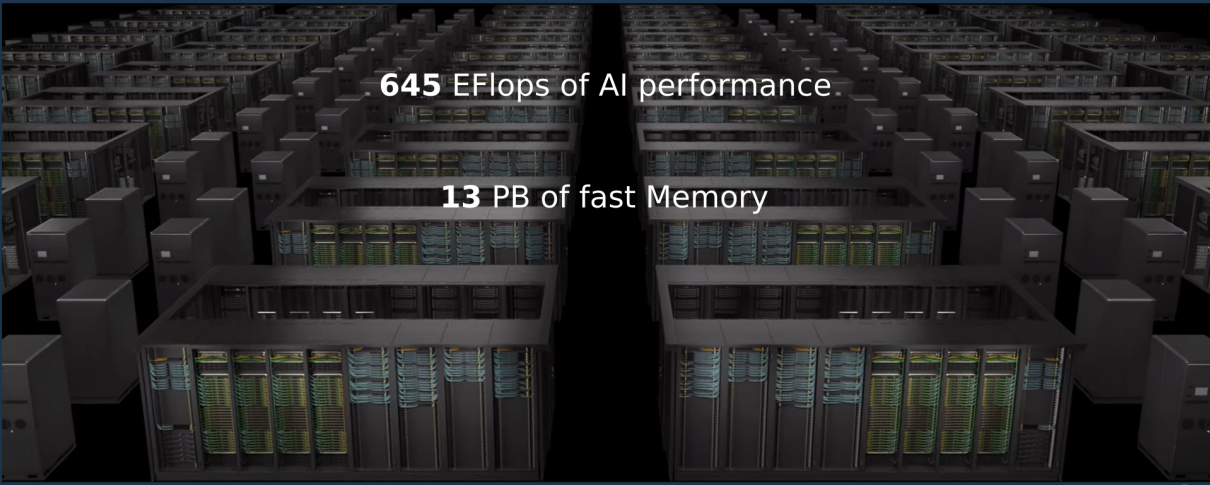
32000 Blackwell GPU

Blackwell Plateform

32000 Blackwell GPU

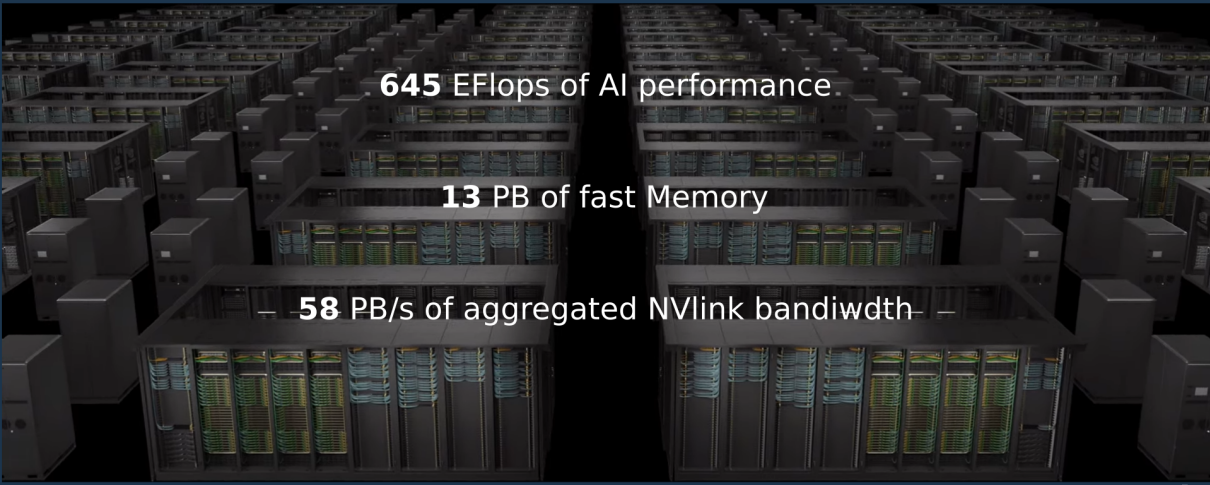
Blackwell Plateform

32000 Blackwell GPU



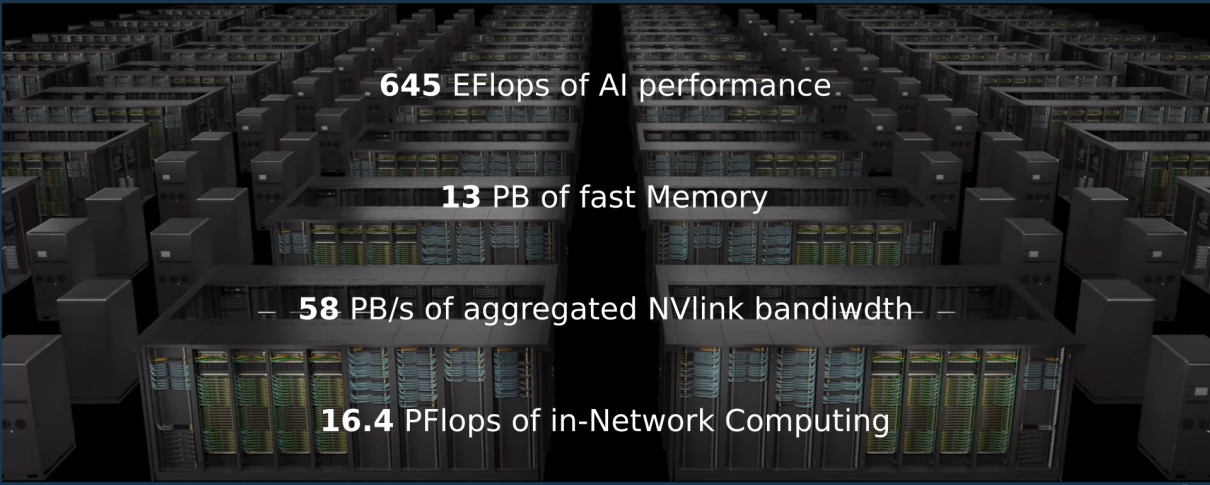
Blackwell Plateform

32000 Blackwell GPU



Blackwell Plateform

32000 Blackwell GPU



645 EFlops of AI performance

13 PB of fast Memory

58 PB/s of aggregated NVlink bandiwdth

16.4 PFlops of in-Network Computing

Photo-lithography

Photo-lithography

Reverse Lithography

Photo-lithography

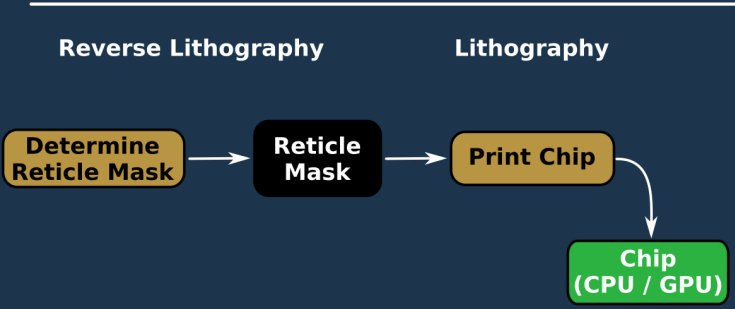


Photo-lithography

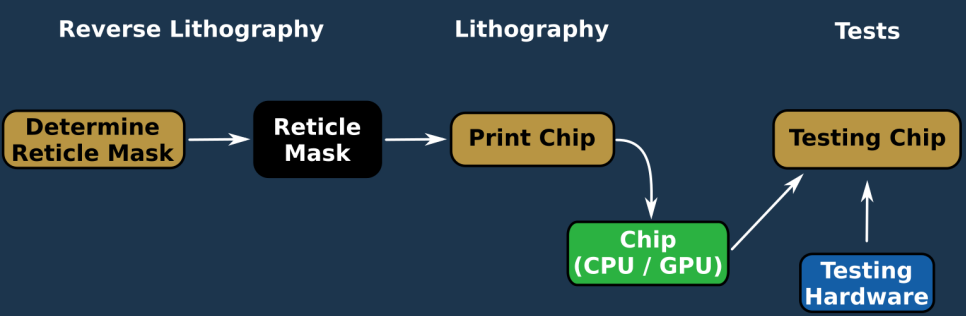


Photo-lithography

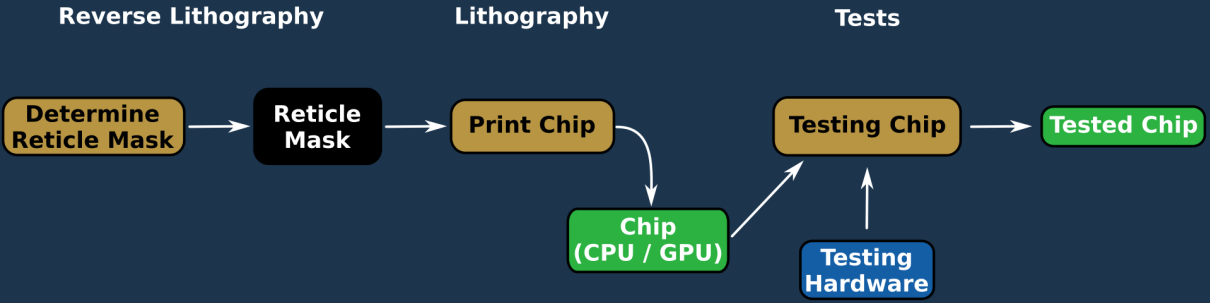


Photo-lithography

Reverse Lithography | **Lithography** | **Tests**

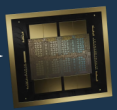
Determine Reticle Mask → **Reticle Mask** → **Print Chip** → **Chip (CPU / GPU)** → **Testing Chip** ← **Testing Hardware** → **Tested Chip**

Computing Centers become chips

Titan (2012)
7MW

20 PFlops

Blackwell (2024)
1 kW



20 PFlops

Photo-lithography

Reverse Lithography

Determine Reticle Mask → **Reticle Mask**

Lithography

Print Chip → **Chip (CPU / GPU)**

Tests

Testing Chip → **Tested Chip**

Testing Hardware

Chip +
Test Components

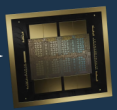
RAS Engine

Computing Centers become chips

Titan (2012)
7MW

20 PFlops

Blackwell (2024)
1 kW



20 PFlops

Photo-lithography

Reverse Lithography

Lithography

Tests

Determine Reticle Mask → Reticle Mask → Print Chip → Testing Chip → Tested Chip

Chip (CPU / GPU)

Testing Hardware

Enable test Monitoring

Chip + Test Components

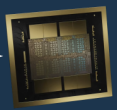
RAS Engine

Computing Centers become chips

Titan (2012)
7MW

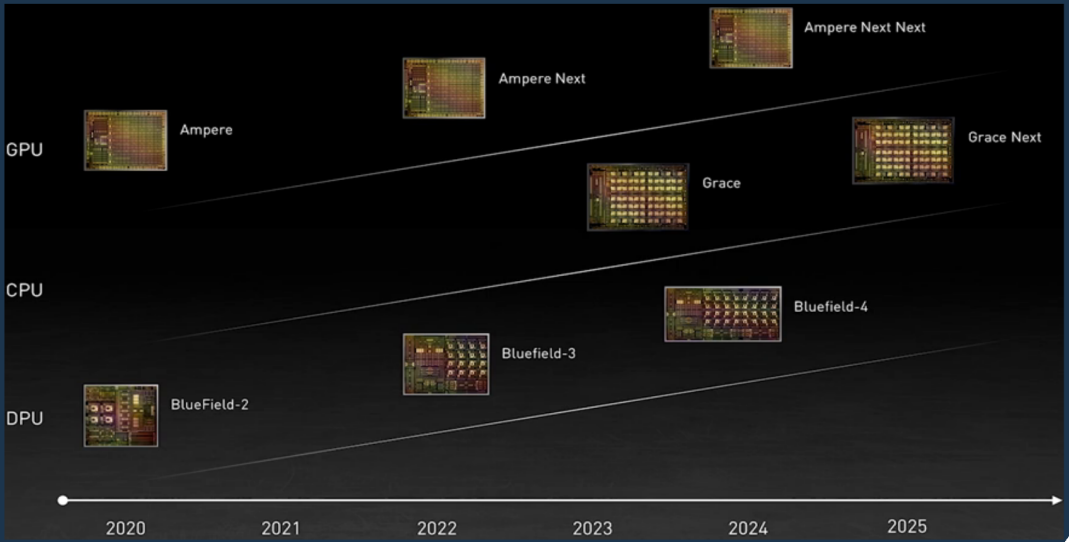
20 PFlops

Blackwell (2024)
1 kW

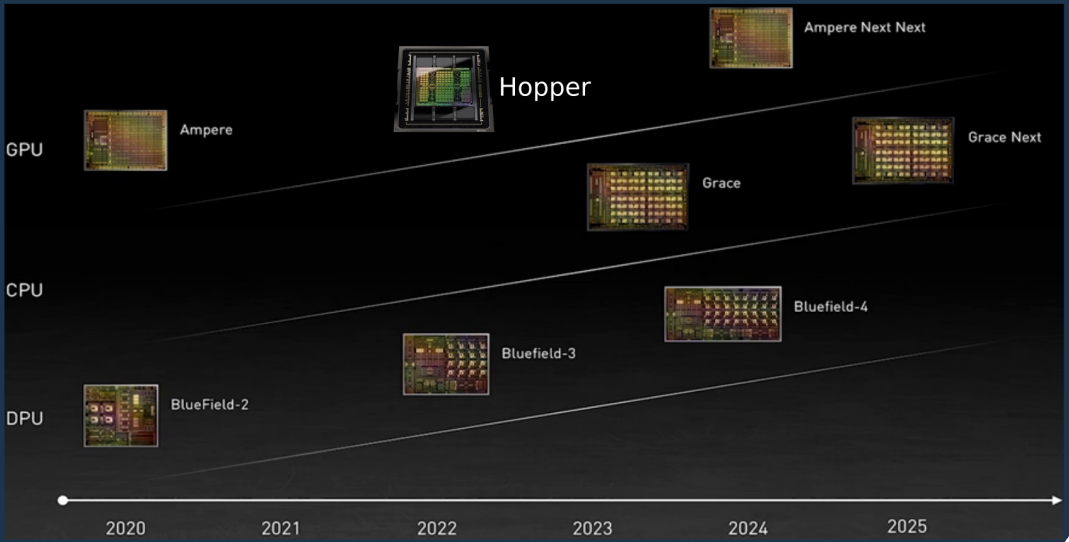


20 PFlops

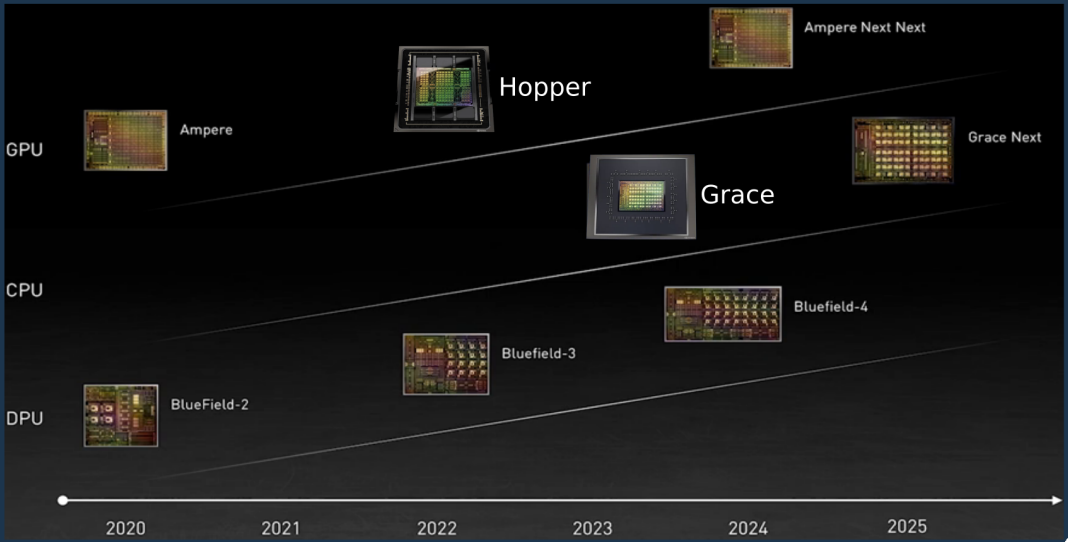
NVidia roadmap



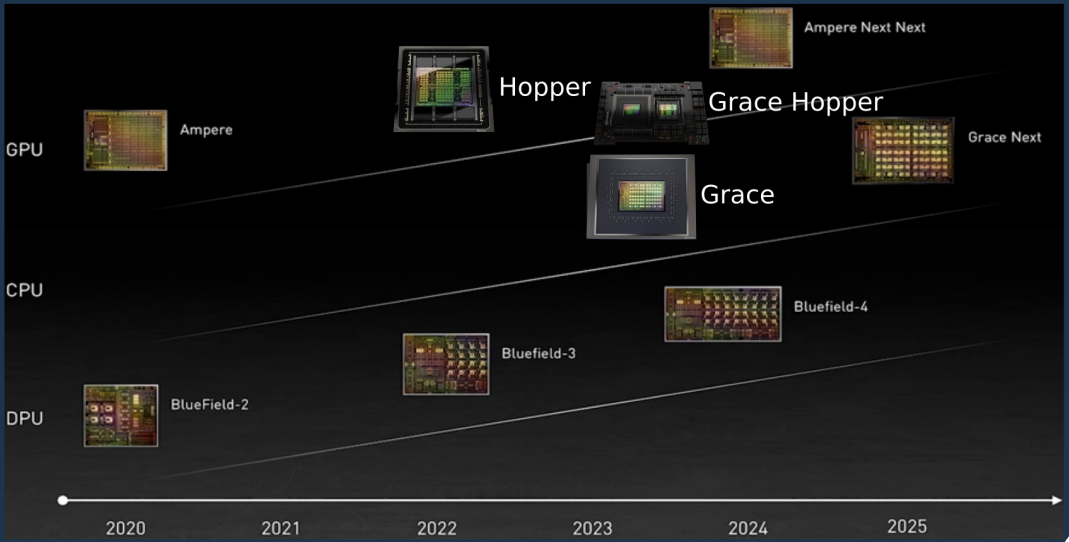
NVidia roadmap



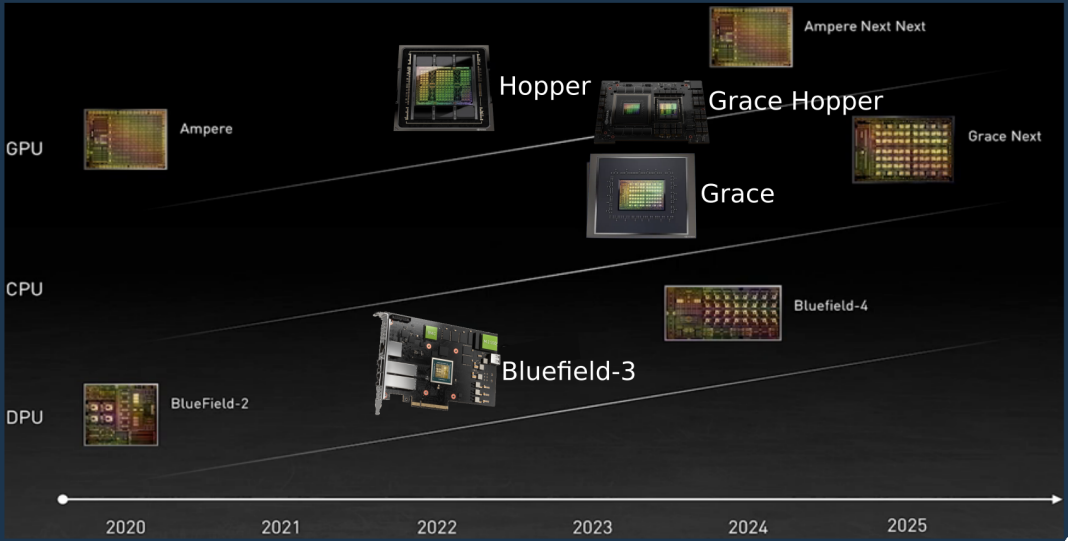
NVidia roadmap



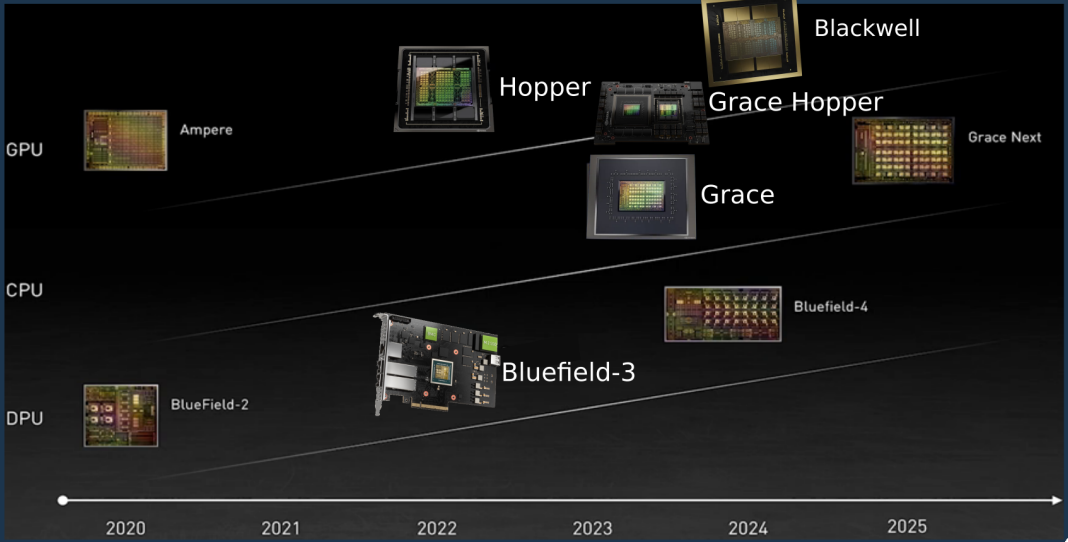
NVidia roadmap



NVidia roadmap

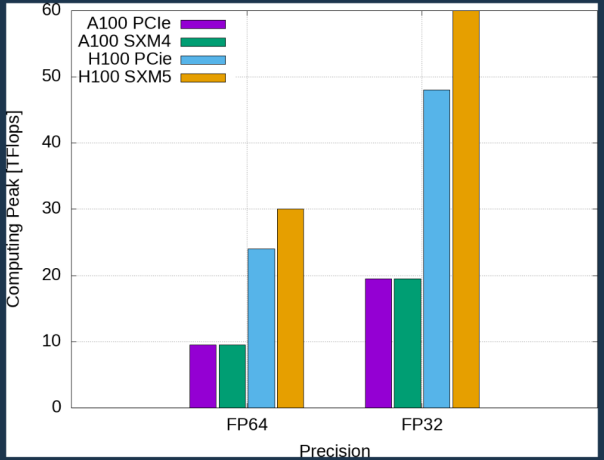


NVidia roadmap

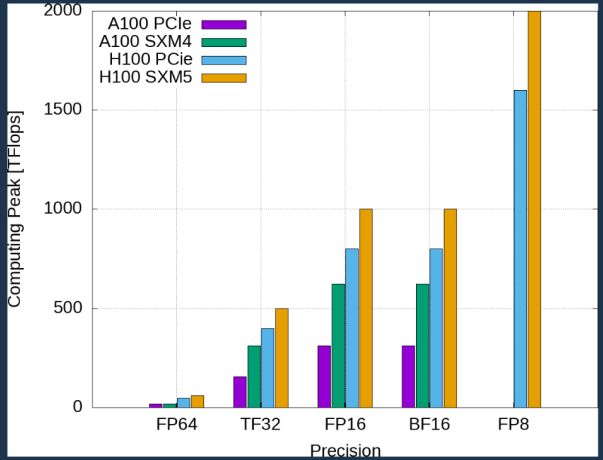


Why using Tensor Cores ?

CUDA Cores

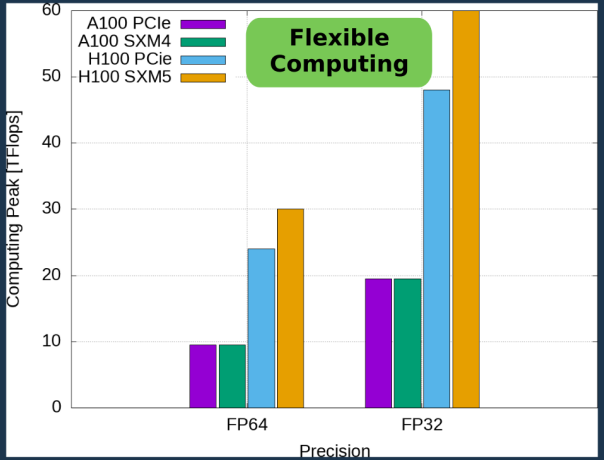


TensorCores

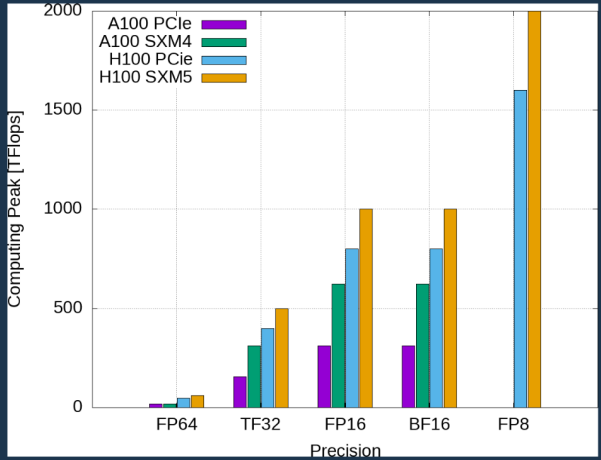


Why using Tensor Cores ?

CUDA Cores

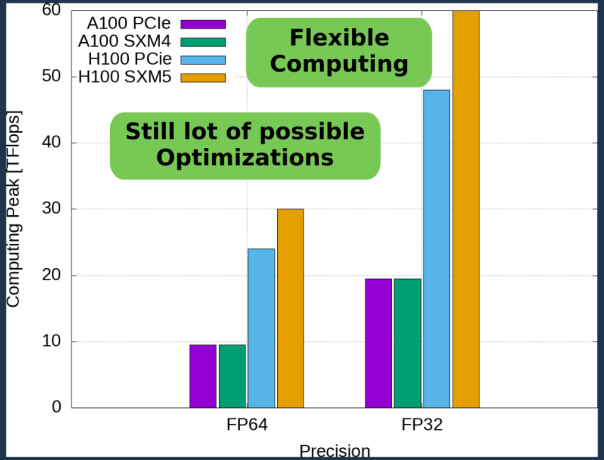


TensorCores

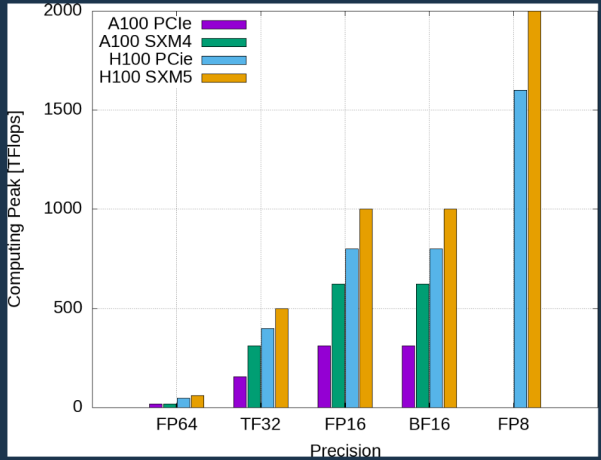


Why using Tensor Cores ?

CUDA Cores

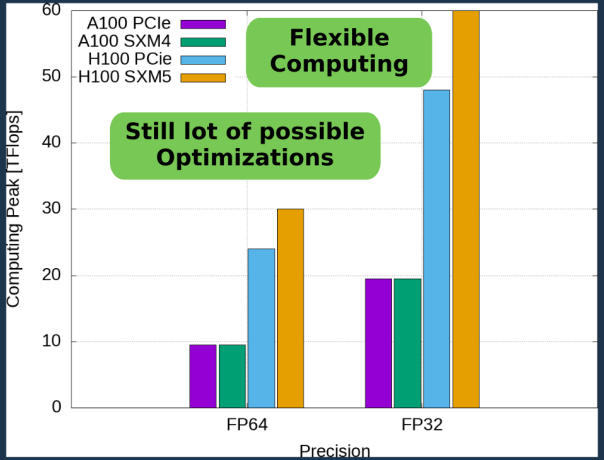


TensorCores

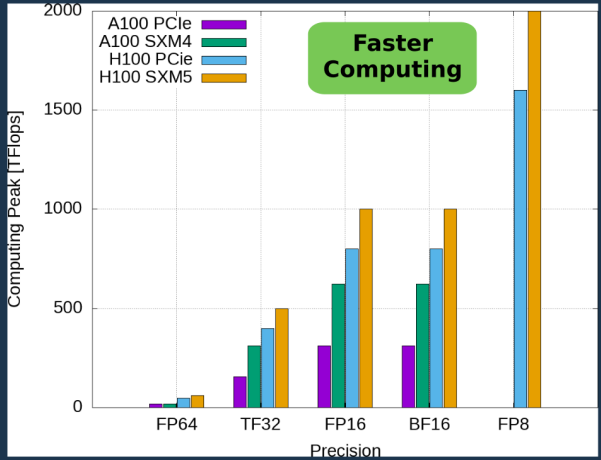


Why using Tensor Cores ?

CUDA Cores

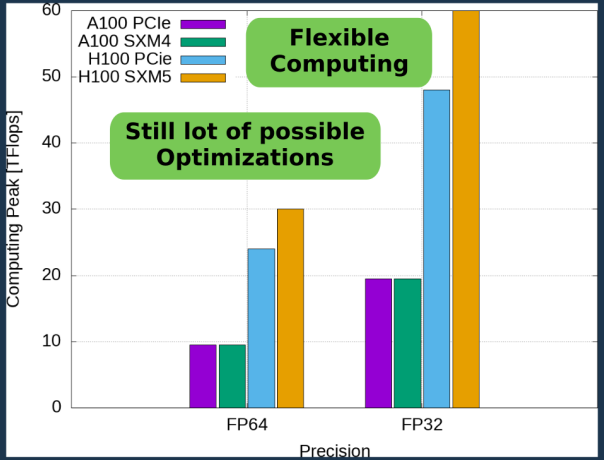


TensorCores

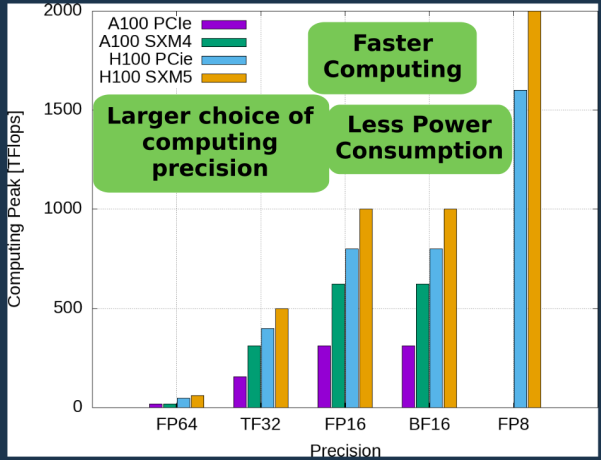


Why using Tensor Cores ?

CUDA Cores

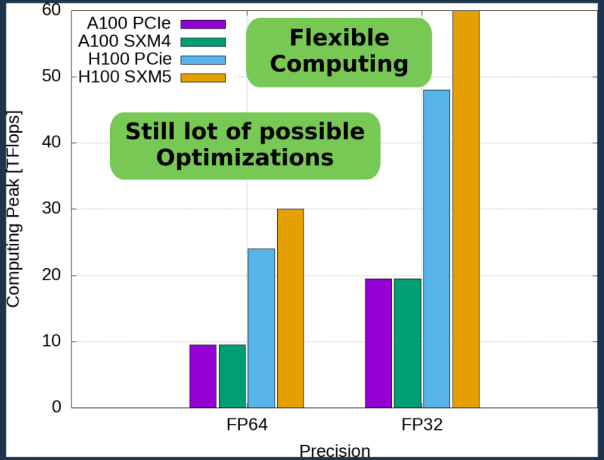


TensorCores

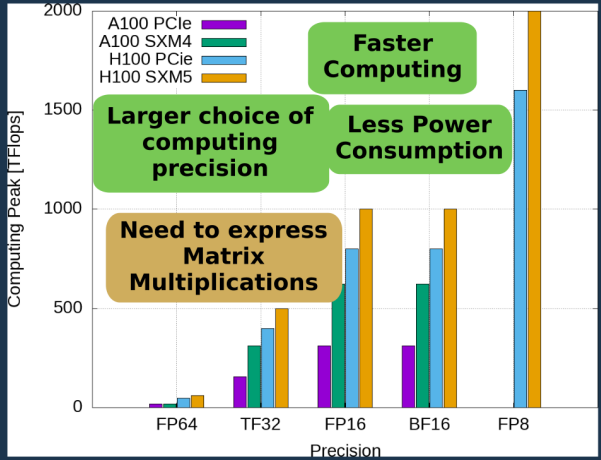


Why using Tensor Cores ?

CUDA Cores

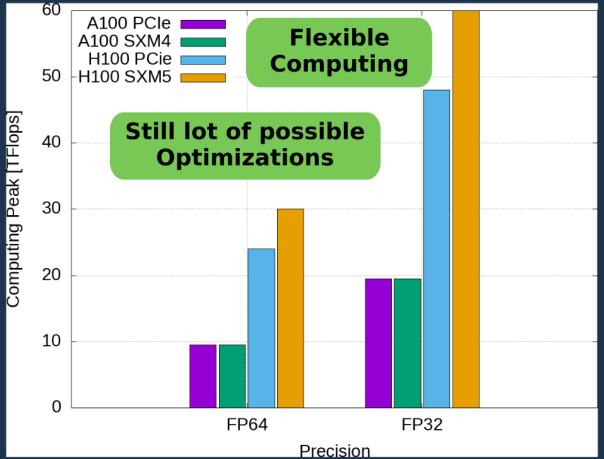


TensorCores

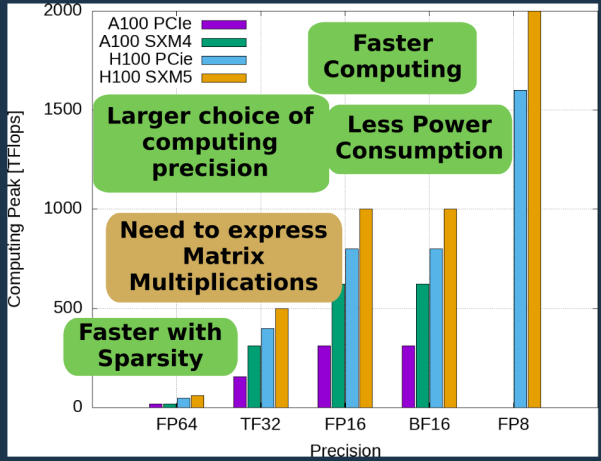


Why using Tensor Cores ?

CUDA Cores

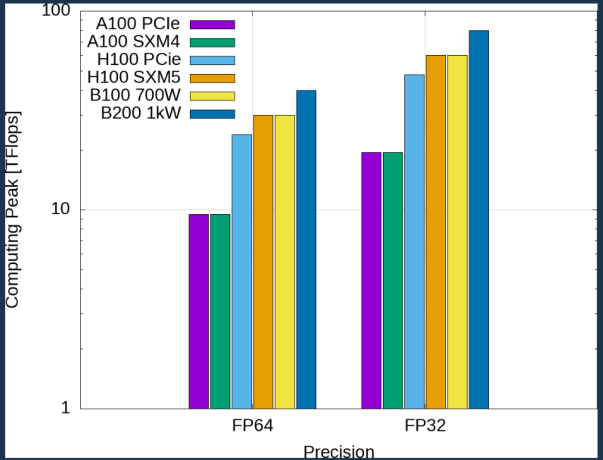


TensorCores

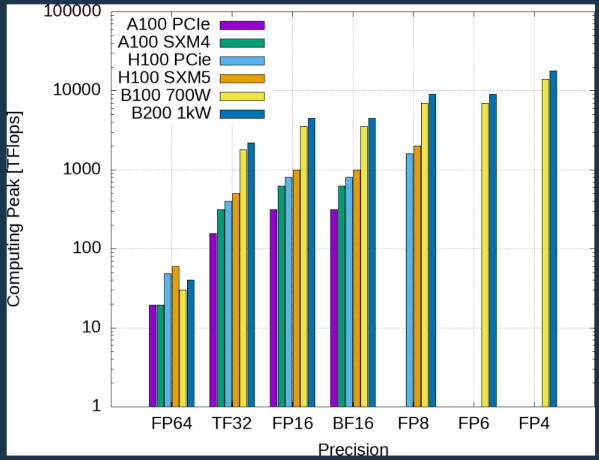


Why using Tensor Cores ?

CUDA Cores

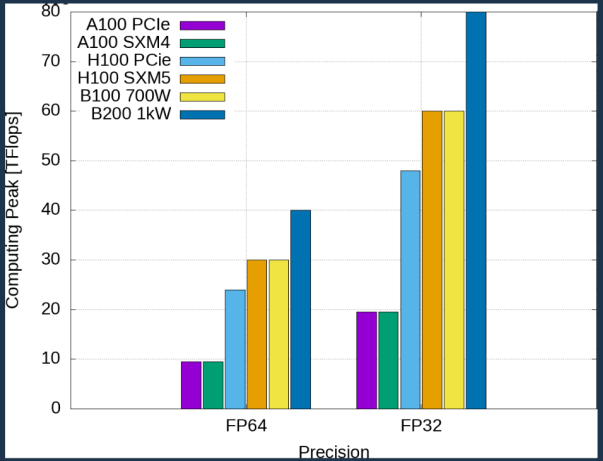


TensorCores

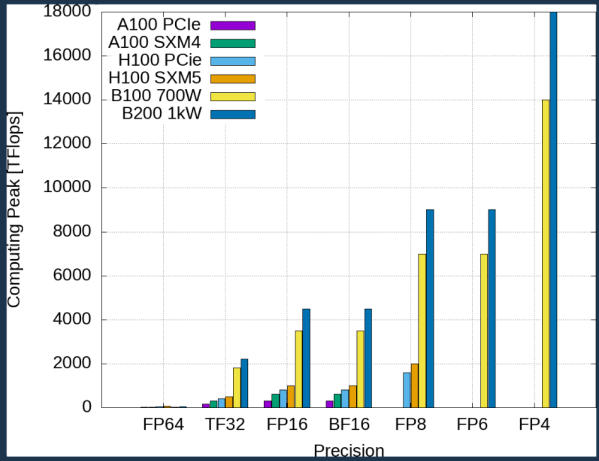


Why using Tensor Cores ?

CUDA Cores



TensorCores



How to use Tensor Cores ?

How to use Tensor Cores ?

Use Existing LIbraries

CUTLASS

BLAS **LAPACK** **PBLAS** **SCALAPACK**

TENSOR **SPARSE** **RAND** **FFT**

cunumeric **legate**

How to use Tensor Cores ?

Use Existing LIbraries

CUTLASS

BLAS | LAPACK | PBLAS | SCALAPACK

TENSOR | SPARSE | RAND | FFT

cunumeric | legate

And libraries based on these ones

How to use Tensor Cores ?

Use Existing LIbraries

CUTLASS

BLAS | LAPACK | PBLAS | SCALAPACK

TENSOR | SPARSE | RAND | FFT

cunumeric | legate

And libraries based on these ones

OK but for the computing precision

How to use Tensor Cores ?

Use Existing LIbraries

CUTLASS

BLAS | LAPACK | PBLAS | SCALAPACK

TENSOR | SPARSE | RAND | FFT

cunumeric | legate

And libraries based on these ones

OK but for the computing precision

LU decomposition
(in **FP16**)

How to use Tensor Cores ?

Use Existing LIbraries

CUTLASS

BLAS | LAPACK | PBLAS | SCALAPACK

TENSOR | SPARSE | RAND | FFT

cunumeric | legate

And libraries based on these ones

OK but for the computing precision

LU decomposition
(in **FP16**)
Mixed precision 6x faster than double with same result

How to use Tensor Cores ?

Use Existing LIbraries

CUTLASS

BLAS **LAPACK** **PBLAS** **SCALAPACK**

TENSOR **SPARSE** **RAND** **FFT**

cunumeric **legate**

And libraries based on these ones

OK but for the computing precision

LU decomposition
(in **FP16**)
Mixed precision 6x faster than double with same result

Ongoing studies on **weather forecasting** (**ALPS**)

How to use Tensor Cores ?

Use Existing LIbraries

CUTLASS

BLAS | LAPACK | PBLAS | SCALAPACK

TENSOR | SPARSE | RAND | FFT

cunumeric | legate

And libraries based on these ones

OK but for the computing precision

LU decomposition
(in **FP16**)
Mixed precision 6x faster than double with same result

Ongoing studies on **weather forecasting** (**ALPS**)

Hardware can **mix automatically** :
- **V100** : **FP32** + **FP16**
- **H100** : **All** available
- **B100** : **All** available
and keep desired precision

CUTLASS

Open Source: https://github.com/NVIDIA/cutlass

CUTLASS

Open Source: https://github.com/NVIDIA/cutlass

GEMM

CUTLASS

Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution**

CUTLASS

Open Source: https://github.com/NVIDIA/cutlass

CUTLASS

Open Source: https://github.com/NVIDIA/cutlass

Precision : From **FP64** to **FP8**

CUTLASS

Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Precision : From **FP64** to **FP8**

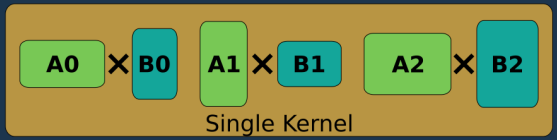
Mixed Input GEMM

CUTLASS

Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Grouped GEMM



Precision : From **FP64** to **FP8**

Mixed Input GEMM

CUTLASS

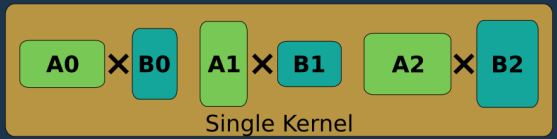
Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Precision : From **FP64** to **FP8**

Mixed Input GEMM

Grouped GEMM



Efficient for **Mixture of Experts**

CUTLASS

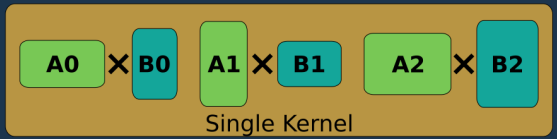
Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Precision : From **FP64** to **FP8**

Mixed Input GEMM

Grouped GEMM



Efficient for **Mixture of Experts**

Composable Load Balancing

CUTLASS

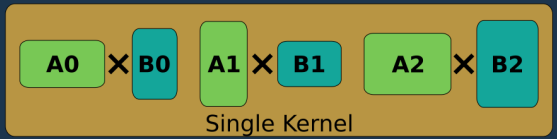
Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Precision : From **FP64** to **FP8**

Mixed Input GEMM

Grouped GEMM



Efficient for **Mixture of Experts**

Composable Load Balancing

Uses Hopper TMA

CUTLASS

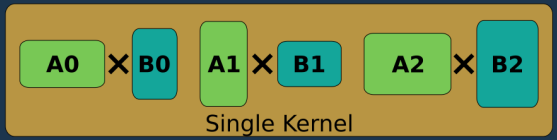
Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Precision : From **FP64** to **FP8**

Mixed Input GEMM

Grouped GEMM



Efficient for **Mixture of Experts**

Composable Load Balancing

Uses Hopper TMA

CUTLASS

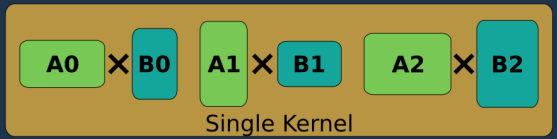
Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Precision : From **FP64** to **FP8**

Mixed Input GEMM

Grouped GEMM



Efficient for **Mixture of Experts**

Composable Load Balancing

Uses Hopper TMA

pip install nvidia-cutlass

CUTLASS

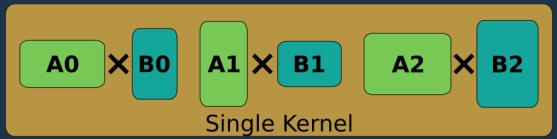
Open Source: https://github.com/NVIDIA/cutlass

GEMM **Convolution** **Reduction**

Precision : From **FP64** to **FP8**

Mixed Input GEMM

Grouped GEMM



Efficient for **Mixture of Experts**

Composable Load Balancing

Uses Hopper TMA

pip install nvidia-cutlass

Computing Backend PyTorch

NSight Profiler Update

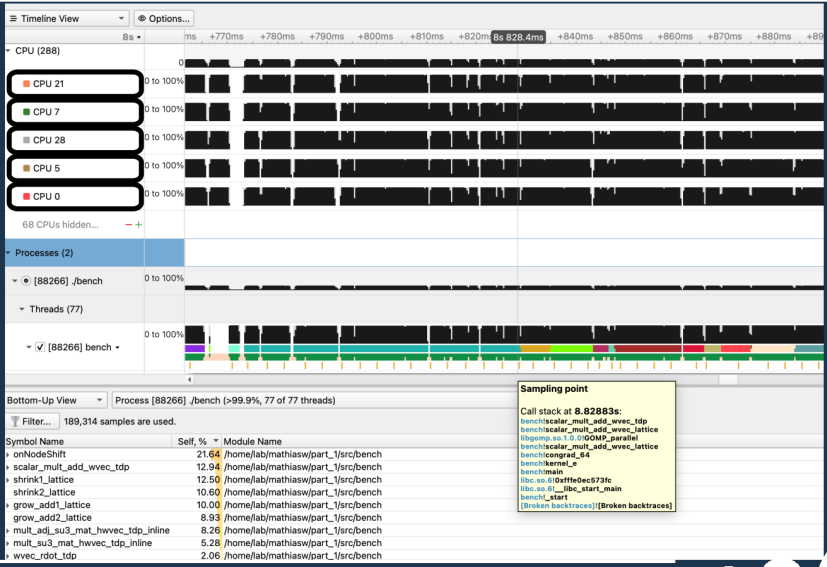
NSight Profiler Update

Laboratoire d'Annecy de Physique des Particules

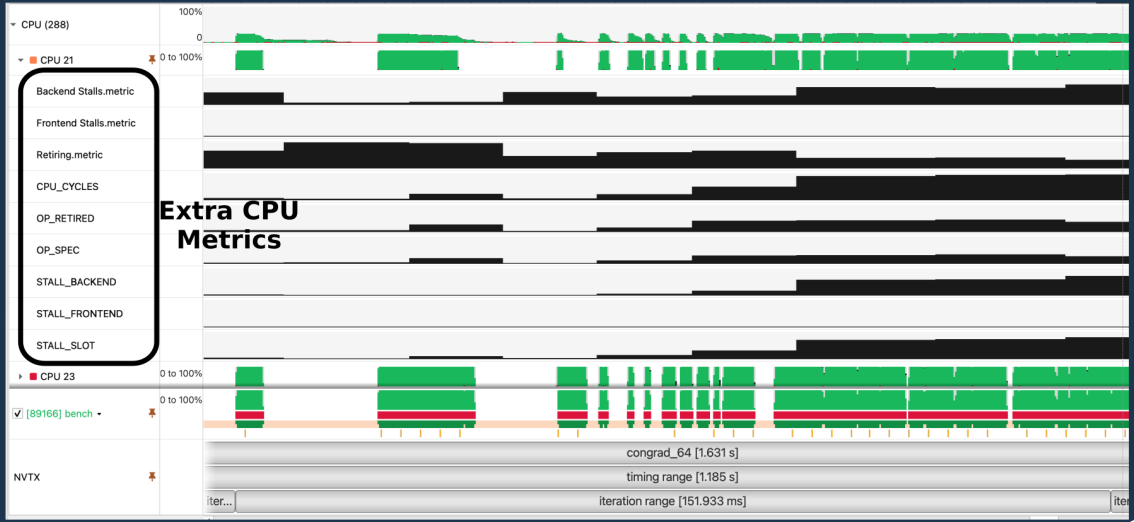
Multi GPU

NSight Profiler Update

Multi CPU



NSight Profiler Update



NSight Profiler Update

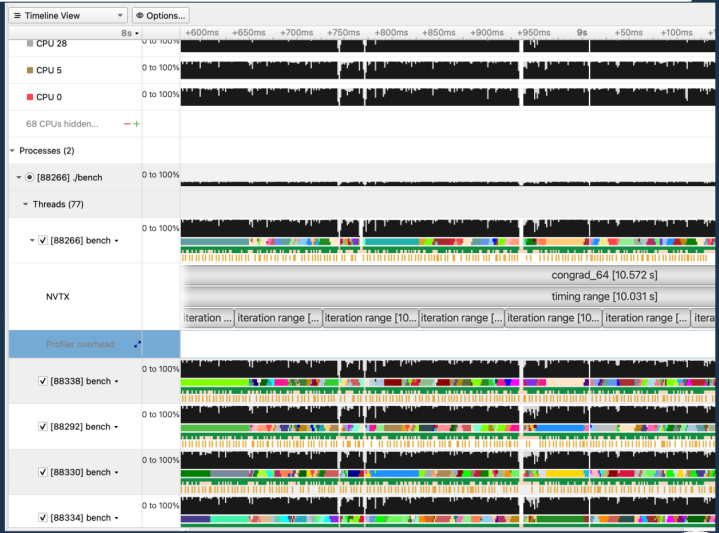
NVTX

Code :
- **Annotation**
- **Provenance**

NSight Profiler Update

NVTX

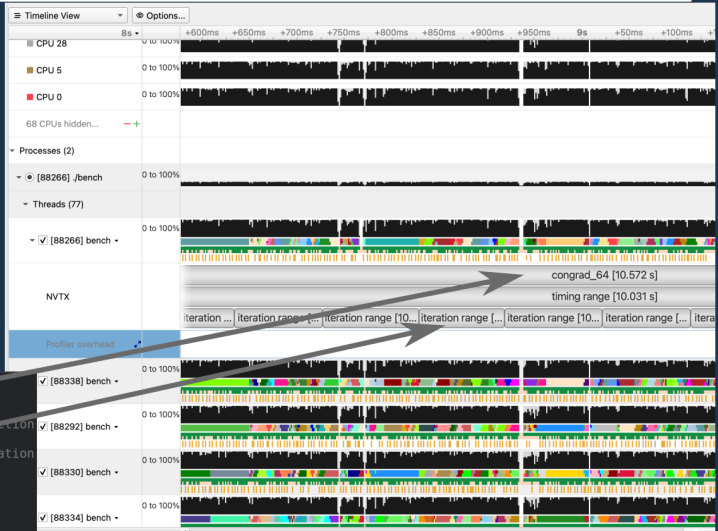
Code :
- **Annotation**
- **Provenance**



NSight Profiler Update

NVTX

Code :
- **Annotation**
- **Provenance**



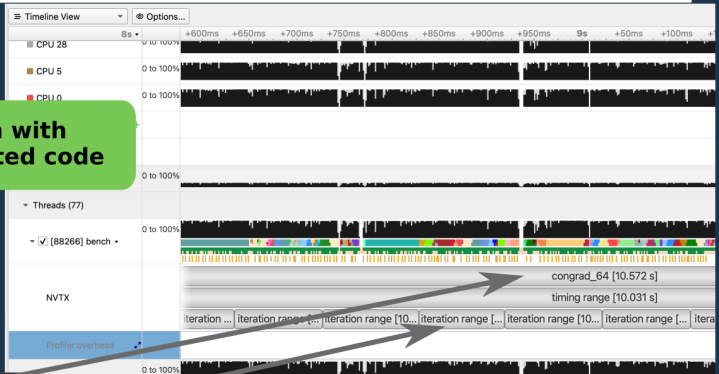

```cpp
#include <nvtx3/nvToolsExt.h>

void congrad_64(){
    nvtxRangePush(__func__); // Range around the whole function
    for(int i = 0; i < 6; ++i) {
        nvtxRangePush("loop range"); // Range for iteration
        // Do ab iteration
        nvtxRangePop(); // End the inner range
    }
    nvtxRangePop(); // End the outer range
}
```

NSight Profiler Update

NVTX

Code :
- **Annotation**
- **Provenance**

Even with generated code




```
1  #include <nvtx3/nvToolsExt.h>
2
3  void congrad_64(){
4  »       nvtxRangePush(__func__); // Range around the whole function
5  »       for(int i = 0; i < 6; ++i) {
6  »       »       nvtxRangePush("loop range"); // Range for iteration
7  »       »       // Do ab iteration
8  »       »       nvtxRangePop(); // End the inner range
9  »       }
10 »       nvtxRangePop(); // End the outer range
11 }
```

CUDA Graph

CUDA Graph

Classic
CUDA Kernel Launch

A

B

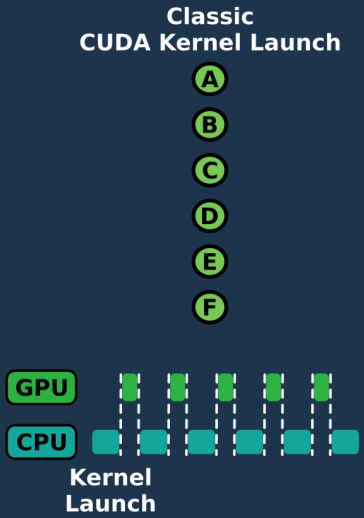
C

D

E

F

CUDA Graph



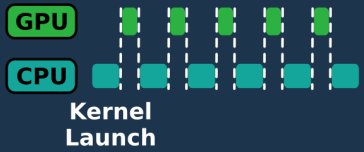
CUDA Graph

Classic
CUDA Kernel Launch

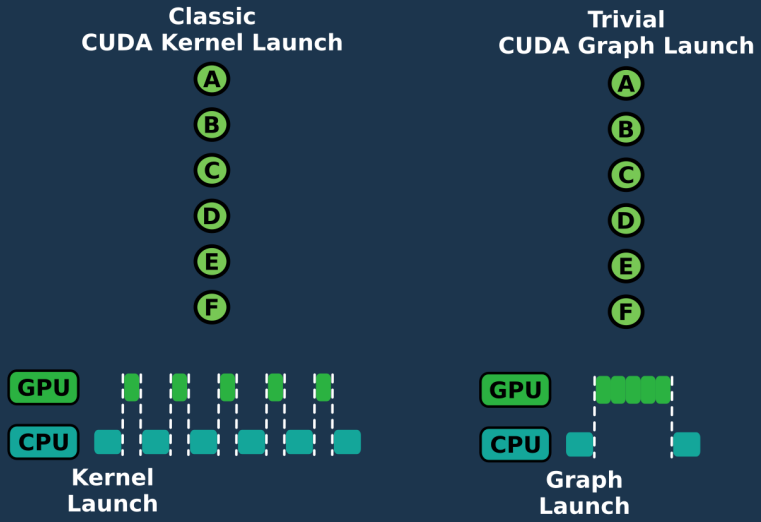
A
B
C
D
E
F

Trivial
CUDA Graph Launch

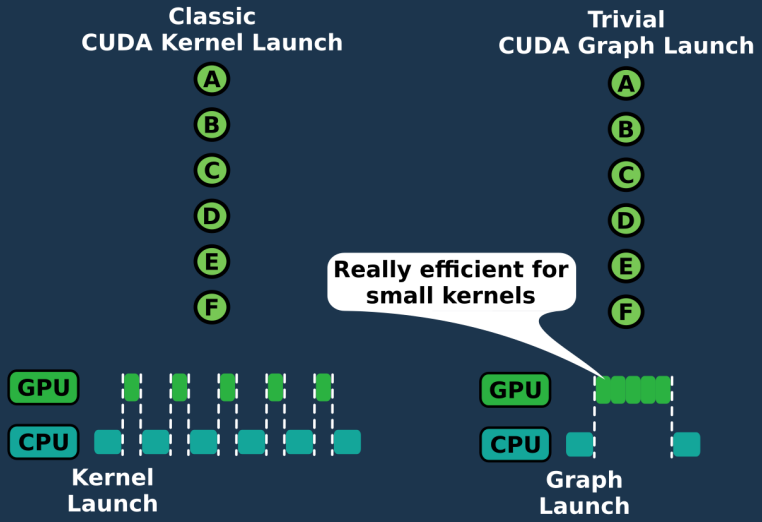
A
B
C
D
E
F



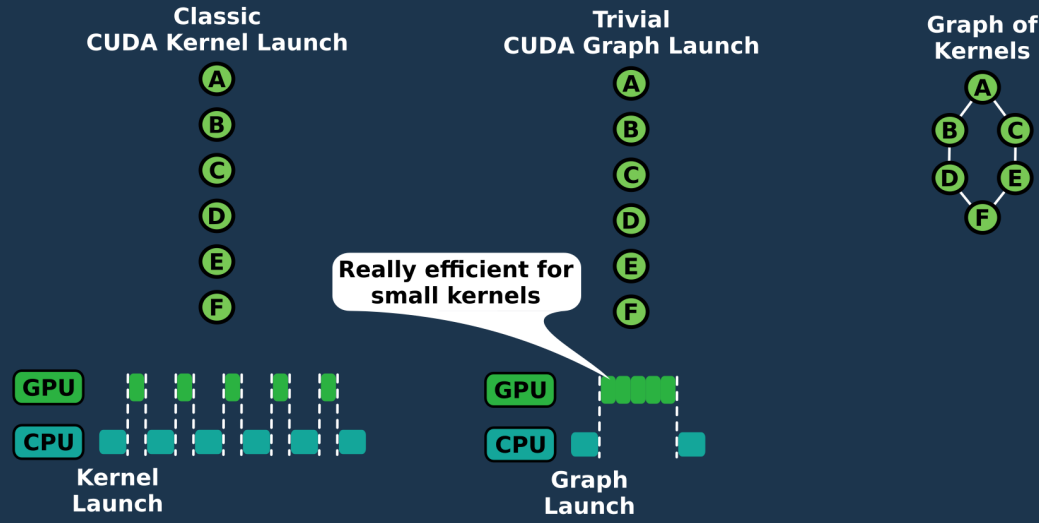
CUDA Graph


Classic
CUDA Kernel Launch
A
B
C
D
E
F
GPU
CPU
Kernel
Launch
Trivial
CUDA Graph Launch
A
B
C
D
E
F
GPU
CPU
Graph
Launch

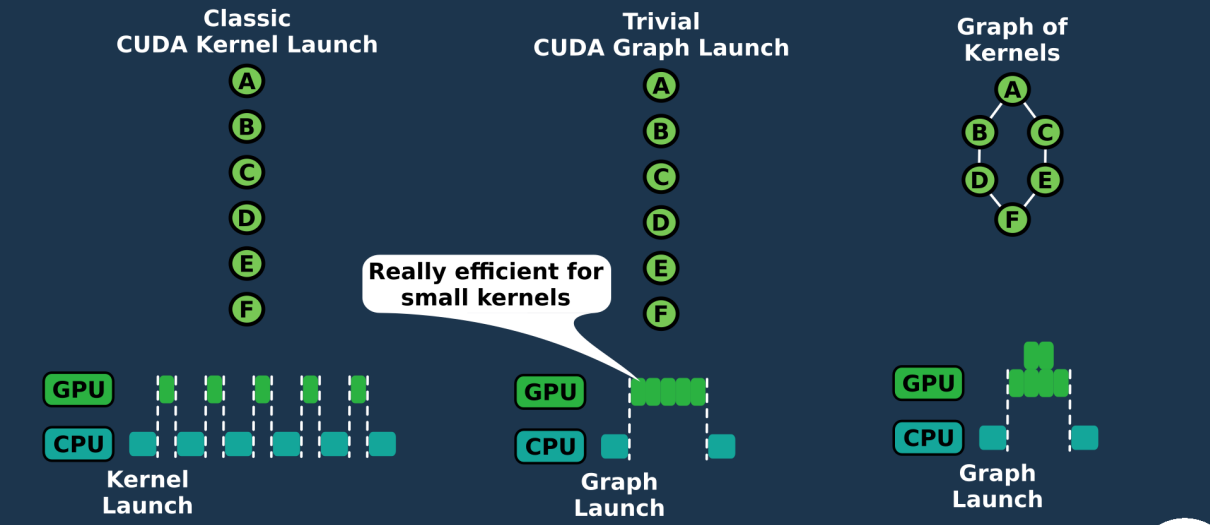
CUDA Graph


Classic
CUDA Kernel Launch
A
B
C
D
E
F
GPU
CPU
Kernel
Launch
Trivial
CUDA Graph Launch
A
B
C
D
E
F
Really efficient for
small kernels
GPU
CPU
Graph
Launch

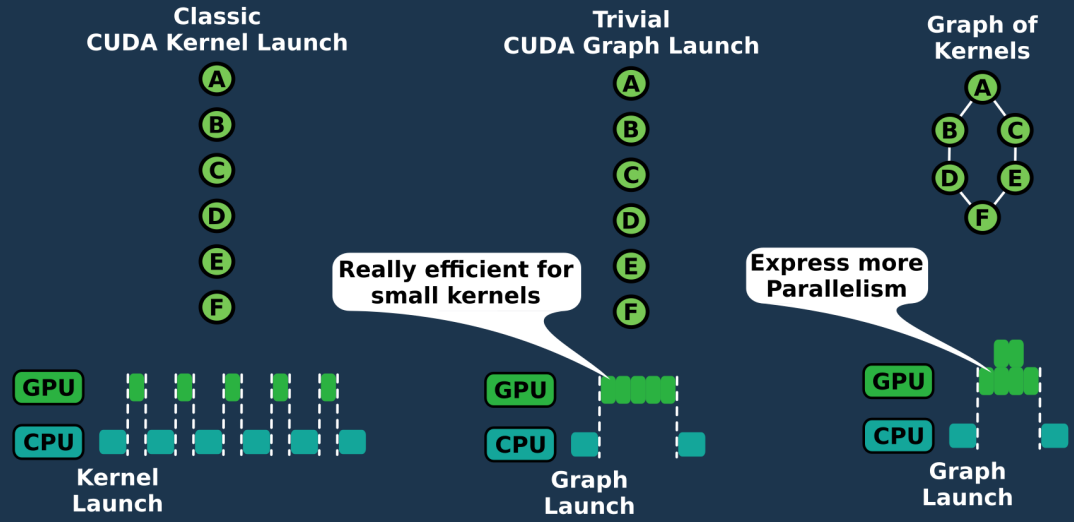
CUDA Graph


Classic
CUDA Kernel Launch
A
B
C
D
E
F
Trivial
CUDA Graph Launch
A
B
C
D
E
F
Graph of
Kernels
A
B
C
D
E
F
Really efficient for
small kernels
GPU
CPU
Kernel
Launch
GPU
CPU
Graph
Launch

CUDA Graph


Classic
CUDA Kernel Launch
A
B
C
D
E
F
Trivial
CUDA Graph Launch
A
B
C
D
E
F
Graph of
Kernels
A
B
C
D
E
F
Really efficient for
small kernels
GPU
CPU
Kernel
Launch
GPU
CPU
Graph
Launch
GPU
CPU
Graph
Launch

CUDA Graph

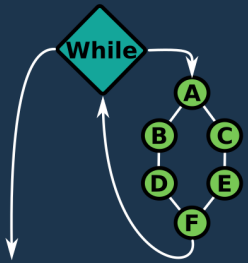

Classic
CUDA Kernel Launch
A
B
C
D
E
F
GPU
CPU
Kernel
Launch
Trivial
CUDA Graph Launch
A
B
C
D
E
F
Really efficient for
small kernels
GPU
CPU
Graph
Launch
Graph of
Kernels
A
B
C
D
E
F
Express more
Parallelism
GPU
CPU
Graph
Launch

CUDA Graph Update

Problem appears again on iterative algorithms

CUDA Graph Update

With Iterative Execution

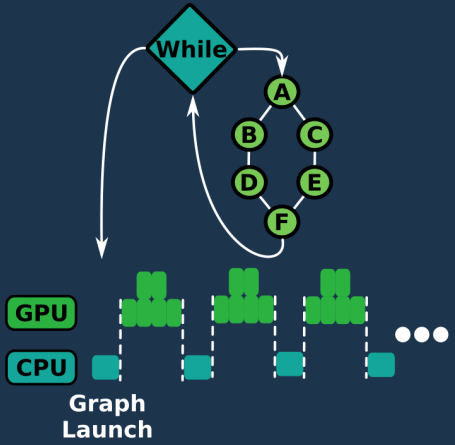


Problem appears again on iterative algorithms

CUDA Graph Update

With Iterative Execution

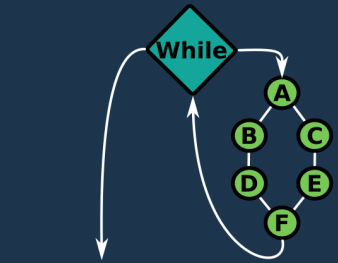
Problem appears again on iterative algorithms

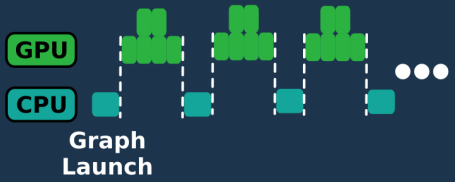


CUDA Graph Update

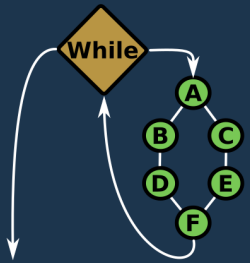
Problem appears again on iterative algorithms

With Iterative Execution

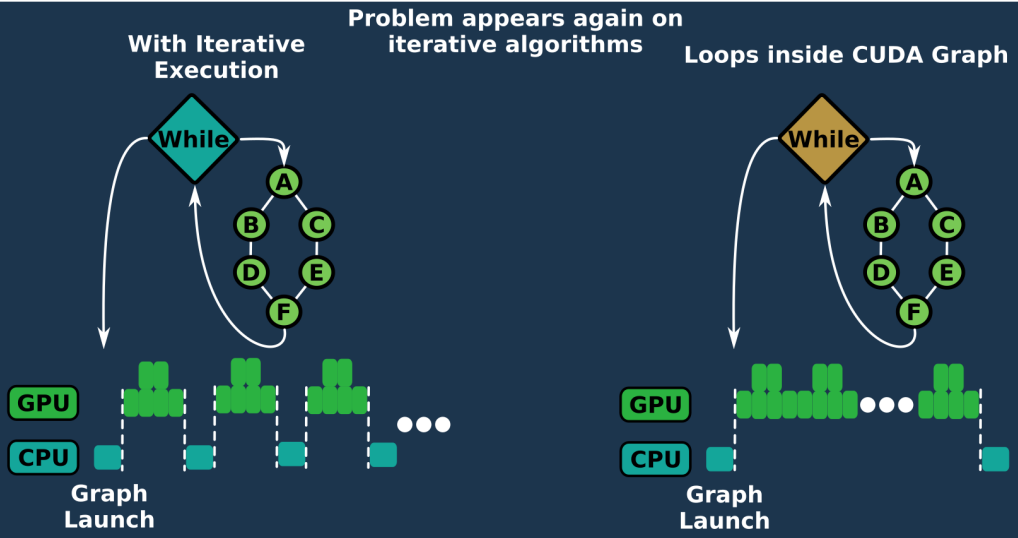




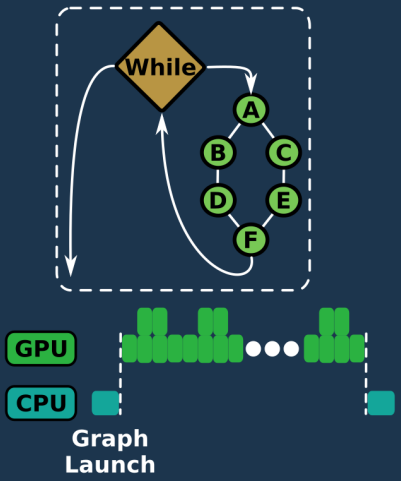
Loops inside CUDA Graph



CUDA Graph Update

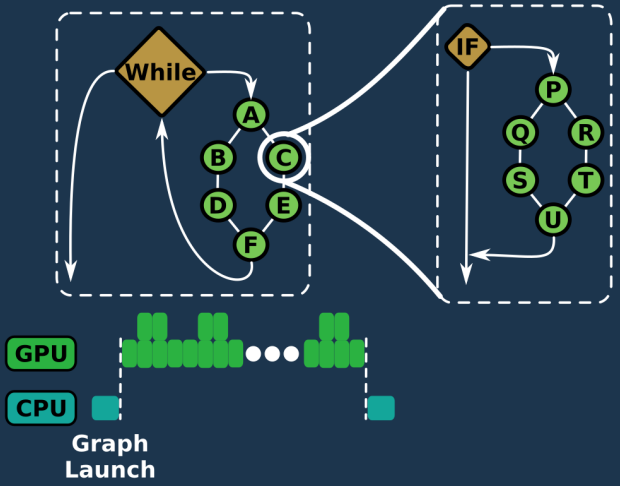


CUDA Graph Update



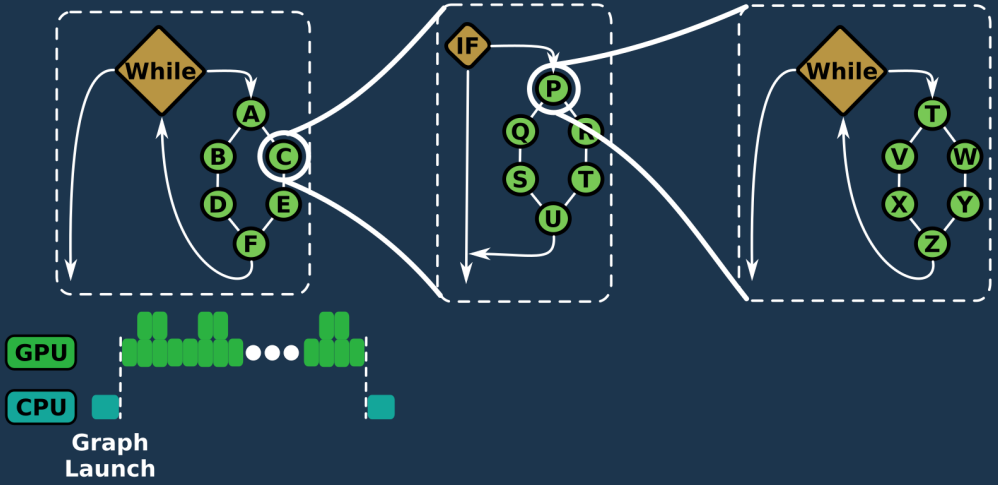
CUDA Graph Update

Sub Graphs, Loops and Conditions



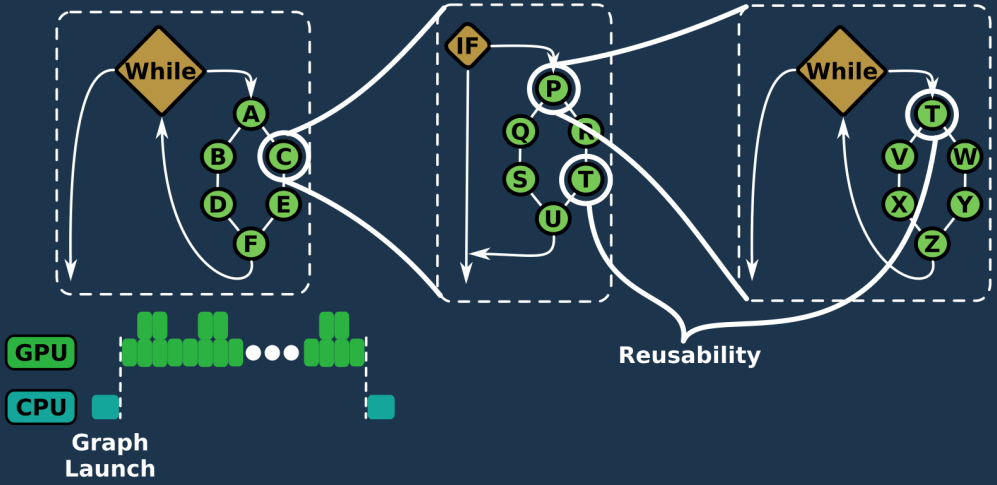
CUDA Graph Update

Sub Graphs, Loops and Conditions



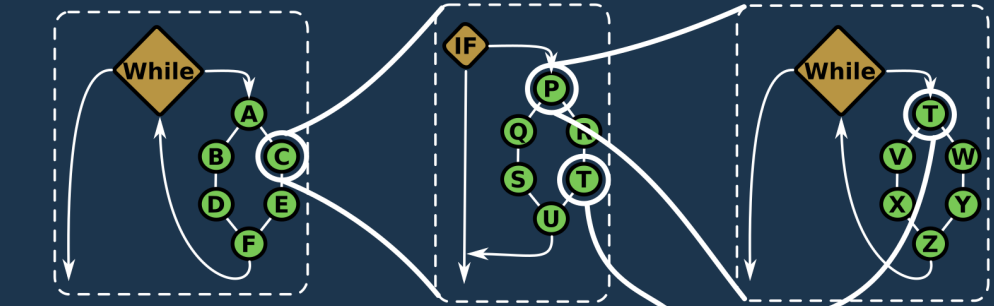
CUDA Graph Update

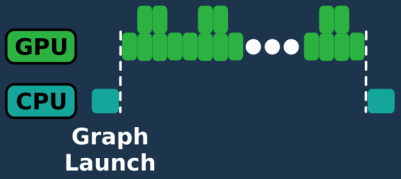
Sub Graphs, Loops and Conditions



CUDA Graph Update

Sub Graphs, Loops and Conditions



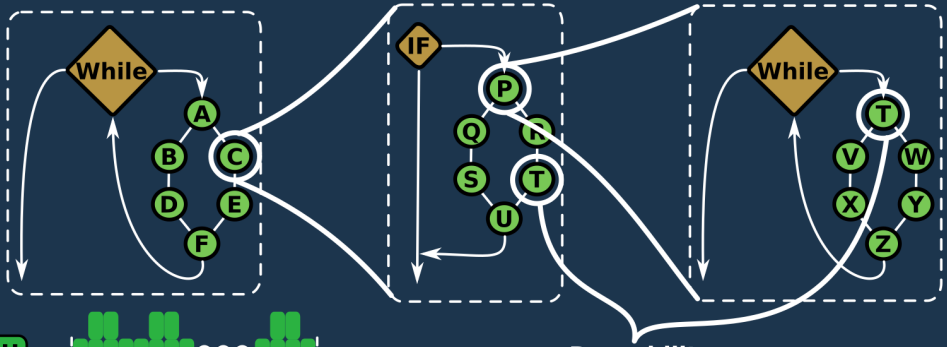


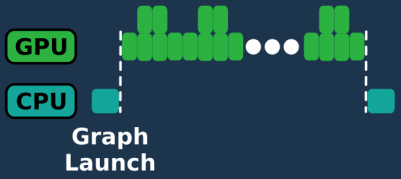
Reusability

cuFFT, NCCL, NVShmem, ...
can be used with CUDA Graphs

CUDA Graph Update

Sub Graphs, Loops and Conditions





Reusability

cuFFT, NCCL, NVShmem, ...
can be used with CUDA Graphs

Available in **CUDA 12.4**

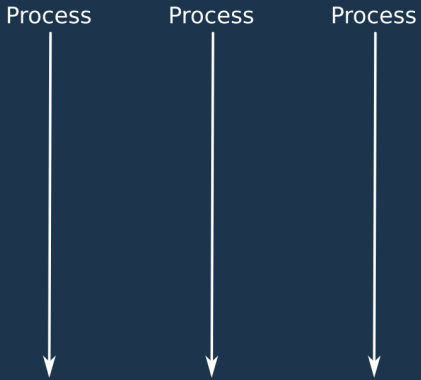
DPU and MVAPICH 2

DPU and MVAPICH 2

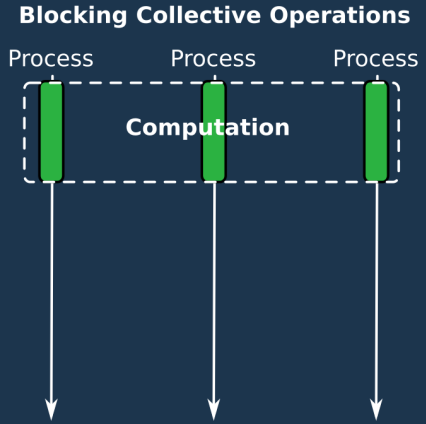
Blocking Collective Operations

DPU and MVAPICH 2

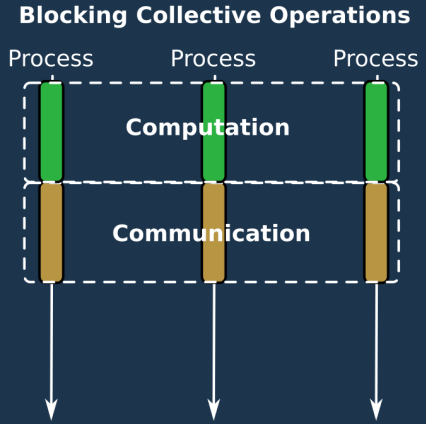
Blocking Collective Operations



DPU and MVAPICH 2

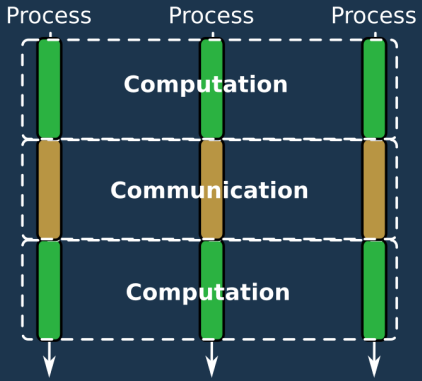


DPU and MVAPICH 2

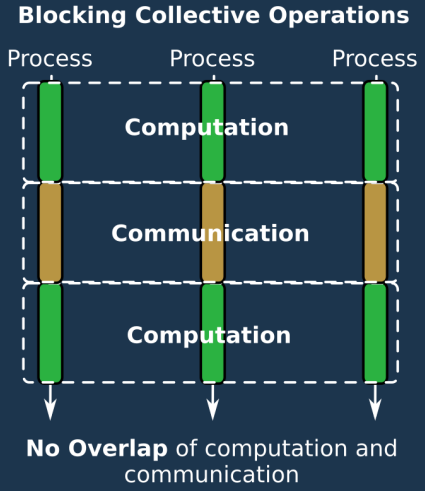


DPU and MVAPICH 2

Blocking Collective Operations

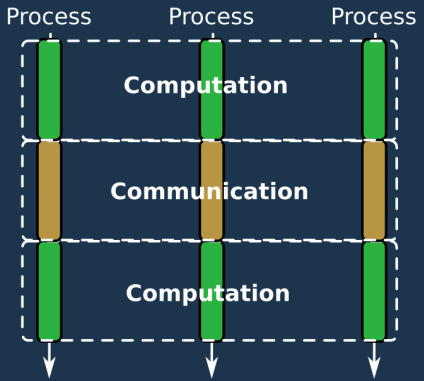


DPU and MVAPICH 2



DPU and MVAPICH 2

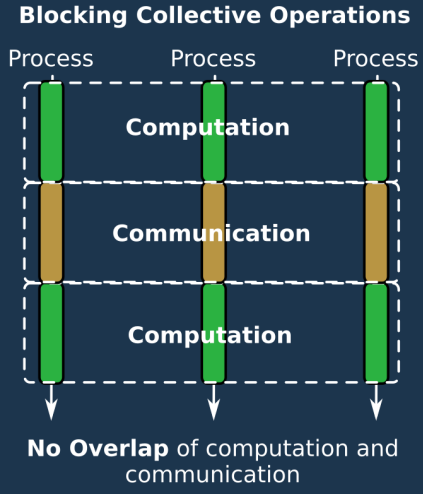
Blocking Collective Operations

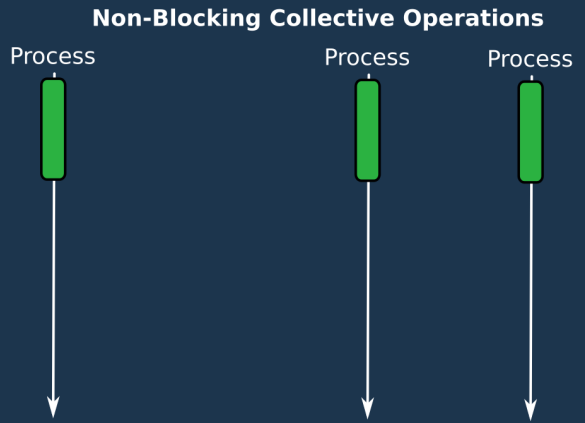


No Overlap of computation and communication

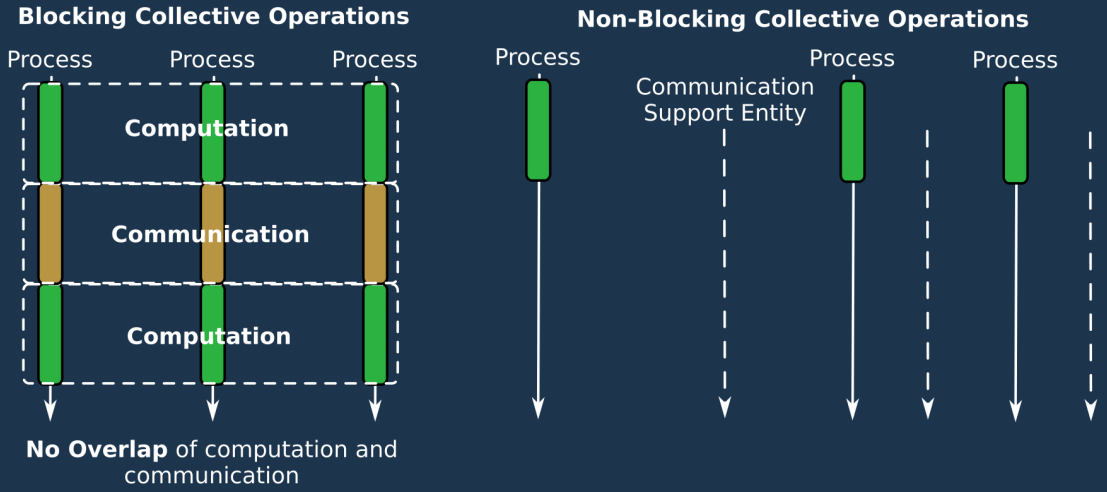
Non-Blocking Collective Operations

DPU and MVAPICH 2

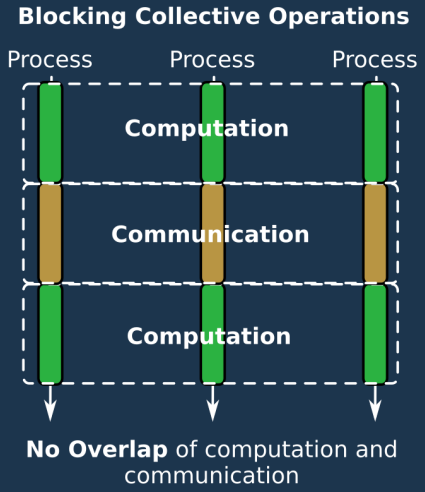


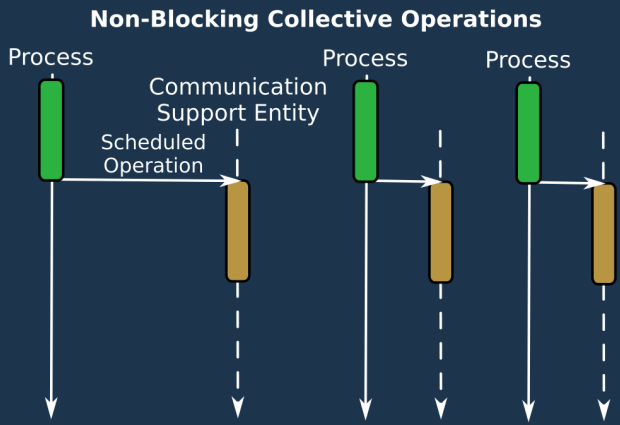


DPU and MVAPICH 2

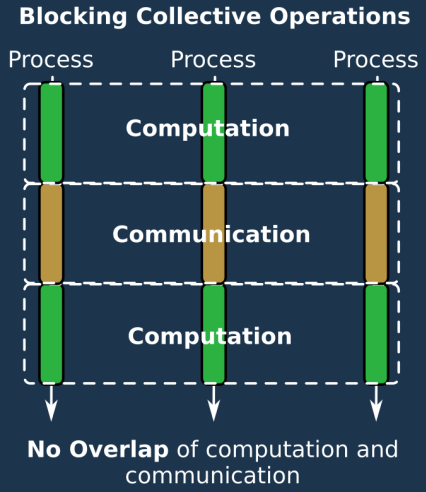


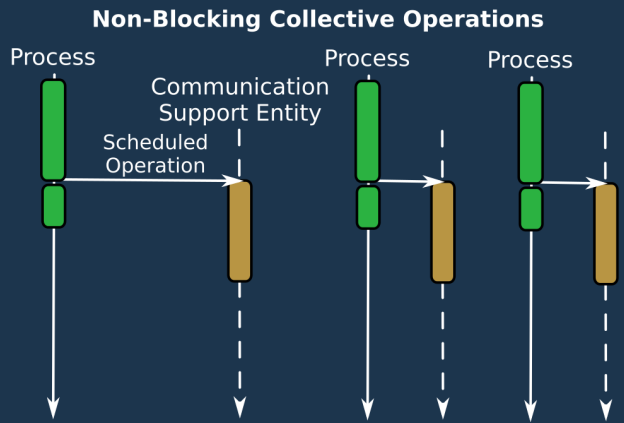
DPU and MVAPICH 2



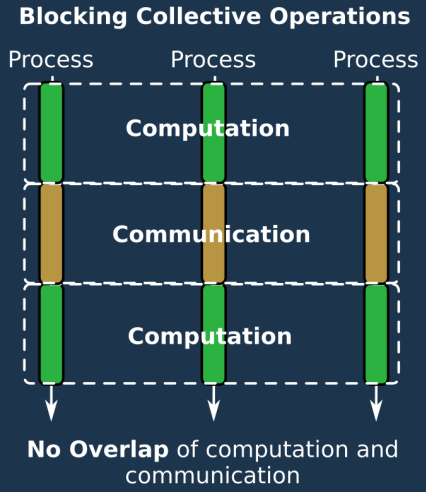


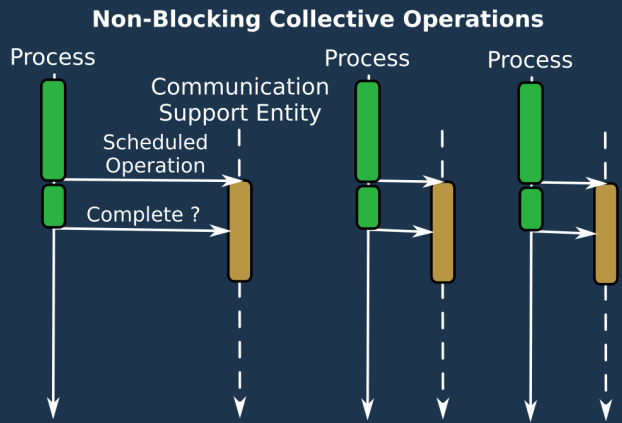
DPU and MVAPICH 2



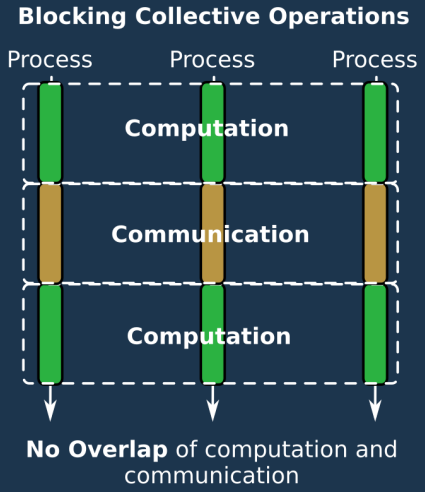


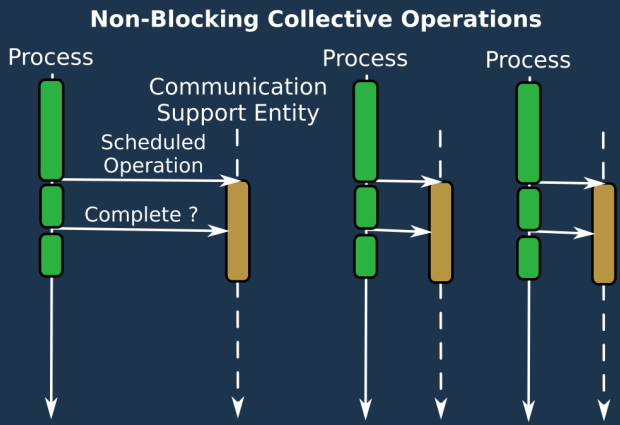
DPU and MVAPICH 2



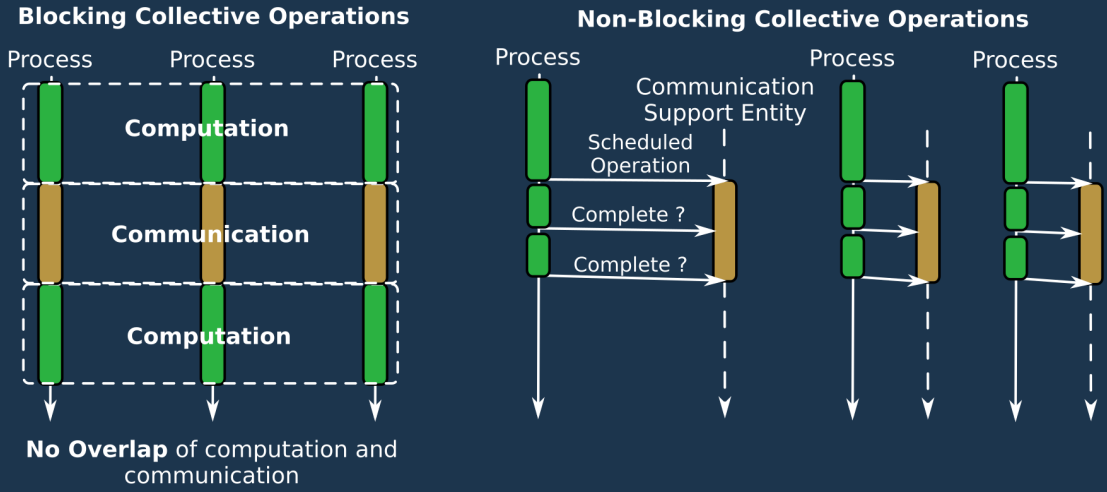


DPU and MVAPICH 2

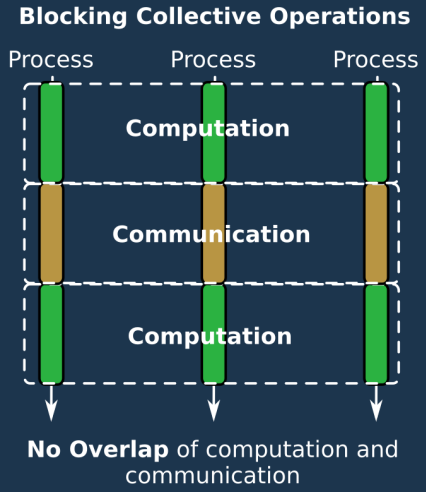


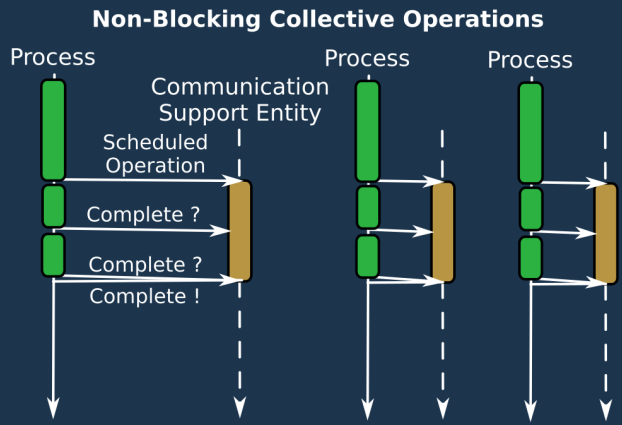


DPU and MVAPICH 2

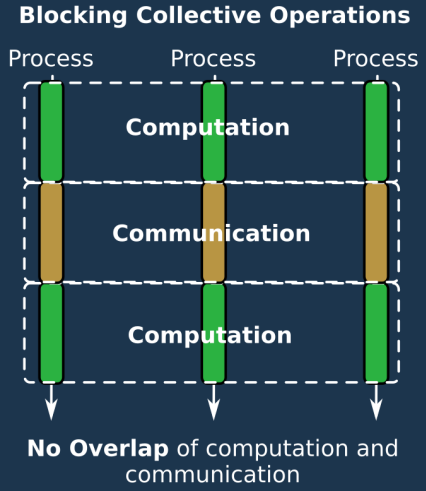


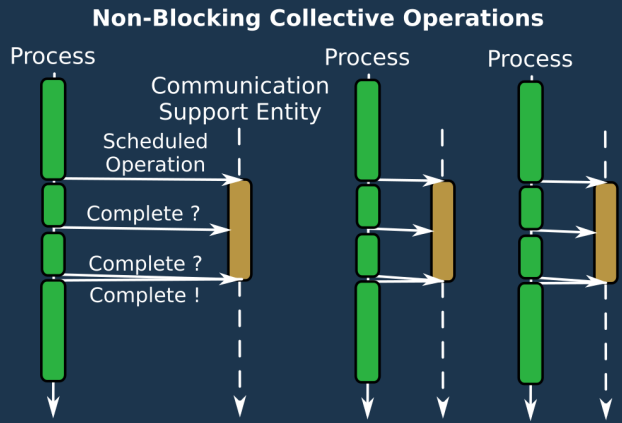
DPU and MVAPICH 2



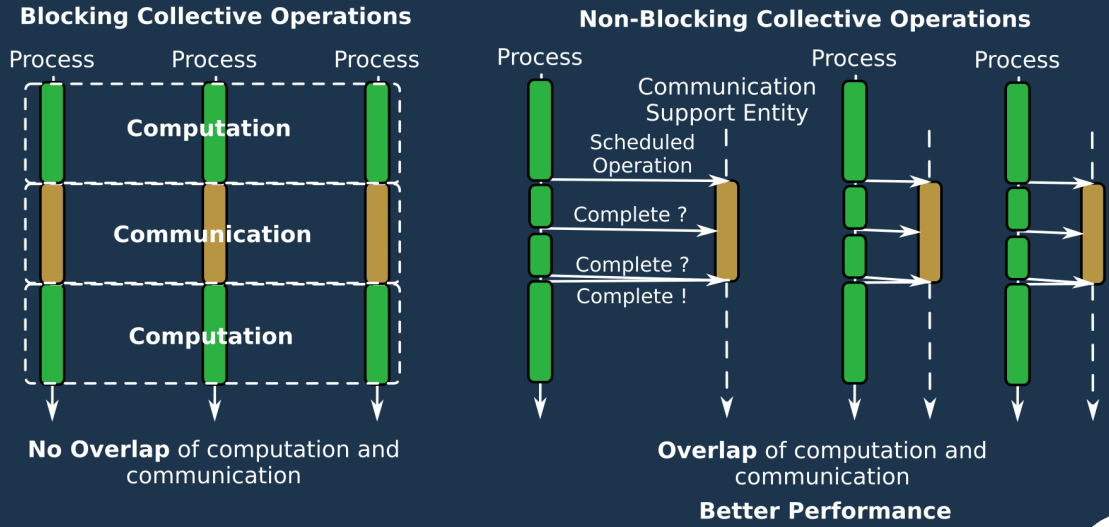


DPU and MVAPICH 2





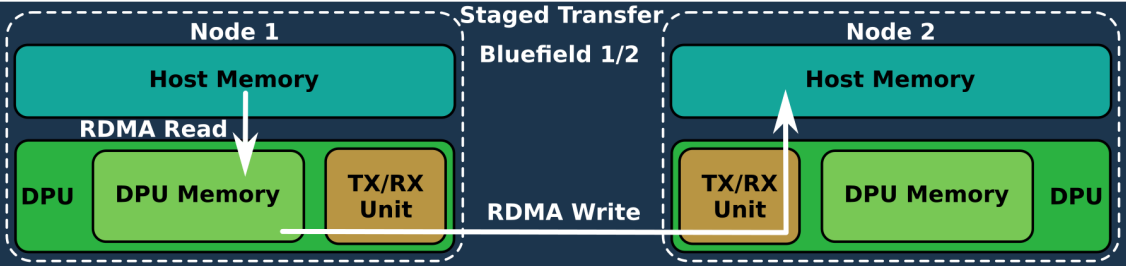
DPU and MVAPICH 2



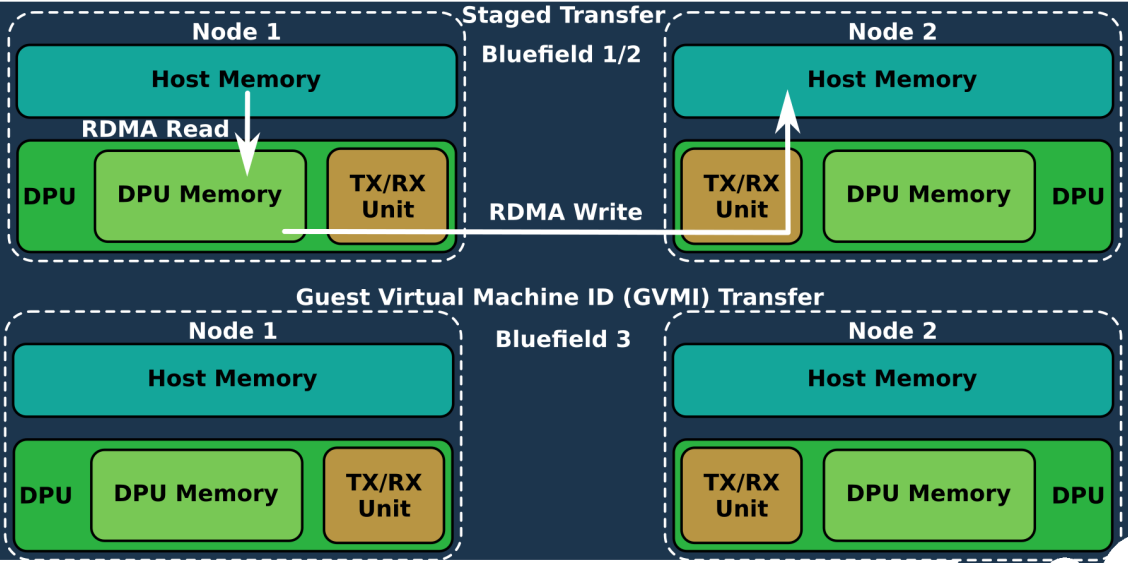
DPU and MVAPICH 2

DPU and MVAPICH 2

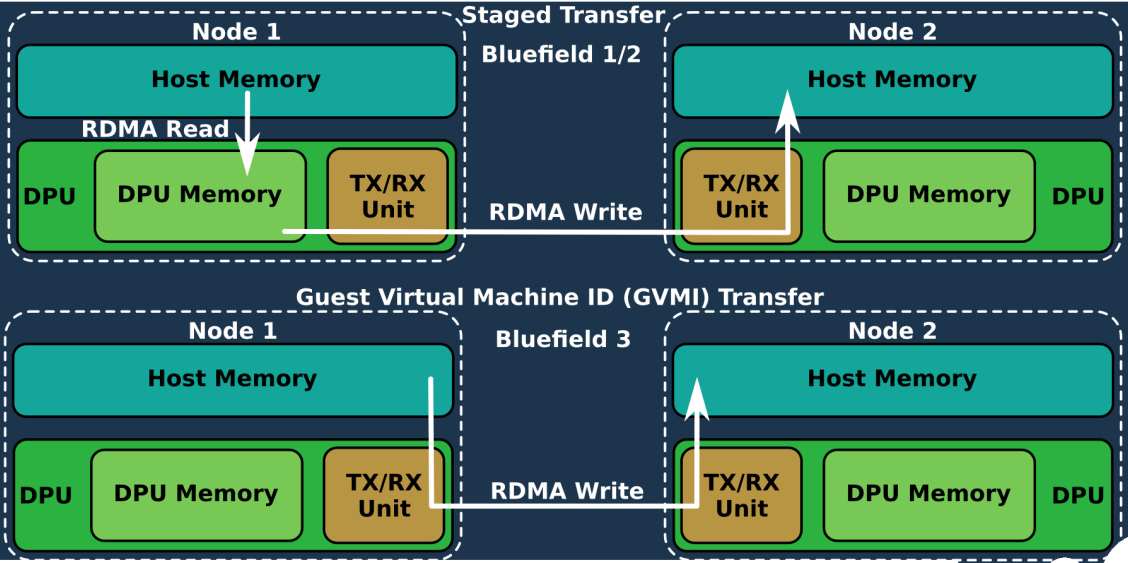
DPU and MVAPICH 2



DPU and MVAPICH 2



DPU and MVAPICH 2



DPU and MVAPICH 2

- Offloads non-blocking **Collectives** to **DPU**
- Offloads non-blocking **Alltoall** (**MPI_Ialltoall**) to **DPU**
- Offloads non-blocking **Broadcast** (**MPI_Ibcast**) to **DPU**
- Offloads **MPI_Isend**/**MPI_Irecv** to **DPU**

Save about **20 %** of **computing** for **all to all** communication

http://mvapich.cse.ohio-state.edu

From **18** to **24%** speed up on 256×256×256 problem from **2** to **8** nodes

DPU and MVAPICH 2

- Offloads non-blocking **Collectives** to **DPU**
- Offloads non-blocking **Alltoall** (**MPI_Ialltoall**) to **DPU**
- Offloads non-blocking **Broadcast** (**MPI_Ibcast**) to **DPU**
- Offloads **MPI_Isend**/**MPI_Irecv** to **DPU**

Save about **20 %** of **computing** for **all to all** communication

http://mvapich.cse.ohio-state.edu

From **18** to **24%** speed up on 256×256×256 problem from **2** to **8** nodes

Offloading **MPI Point-to-Point** and **Reduction** with **PETSc**

https://petsc.org/release/overview/

PETSc used in **Adflow**, **DAFoam**, **FreeFEM**, **MFEM**, **MOOSE**, **OpenFoam**, etc

DPU and MVAPICH 2

- Offloads non-blocking **Collectives** to **DPU**
- Offloads non-blocking **Alltoall** (**MPI_Ialltoall**) to **DPU**
- Offloads non-blocking **Broadcast** (**MPI_Ibcast**) to **DPU**
- Offloads **MPI_Isend/MPI_Irecv** to **DPU**

Save about **20 %** of **computing** for **all to all** communication

http://mvapich.cse.ohio-state.edu

From **18** to **24%** speed up on 256×256×256 problem from **2** to **8** nodes

Offloading **MPI Point-to-Point** and **Reduction** with **PETSc**

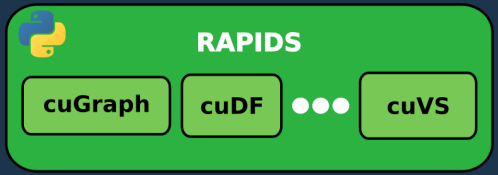
https://petsc.org/release/overview/

PETSc used in **Adflow**, **DAFoam**, **FreeFEM**, **MFEM**, **MOOSE**, **OpenFoam**, etc

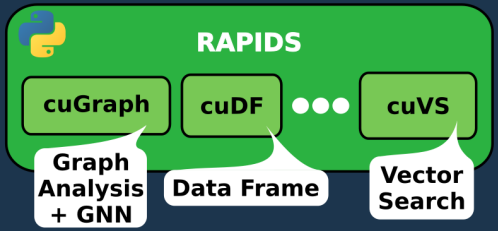
Exploiting **DPUs** for **Deep Neural Network Training** (**Data Augmentation**, **Testing Processes**)

RAPIDS and Legate Frameworks

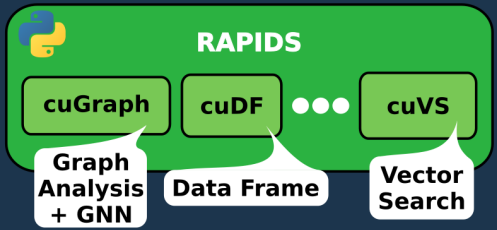
RAPIDS and Legate Frameworks

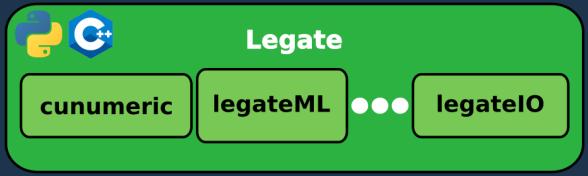


RAPIDS and Legate Frameworks

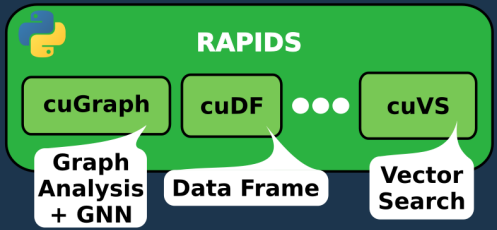


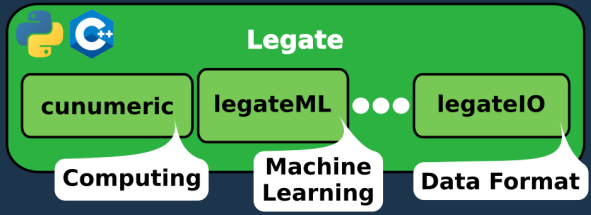
RAPIDS and Legate Frameworks



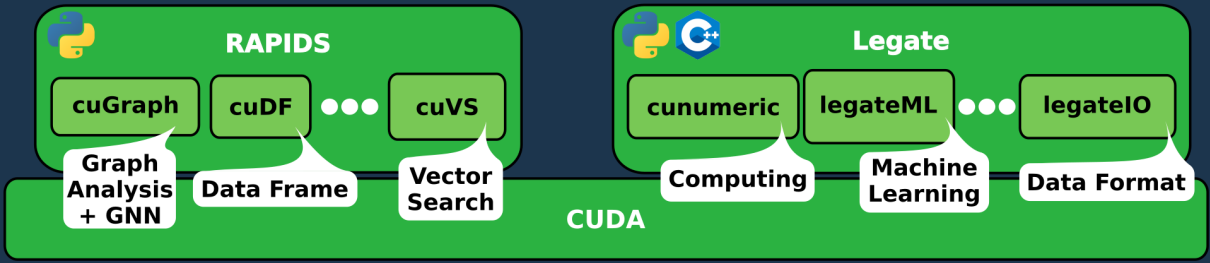


RAPIDS and Legate Frameworks

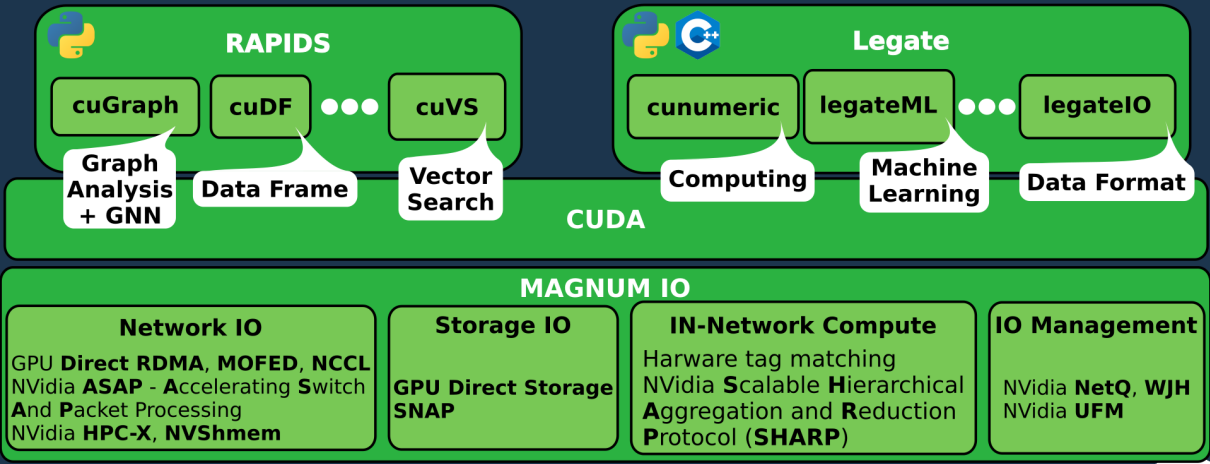




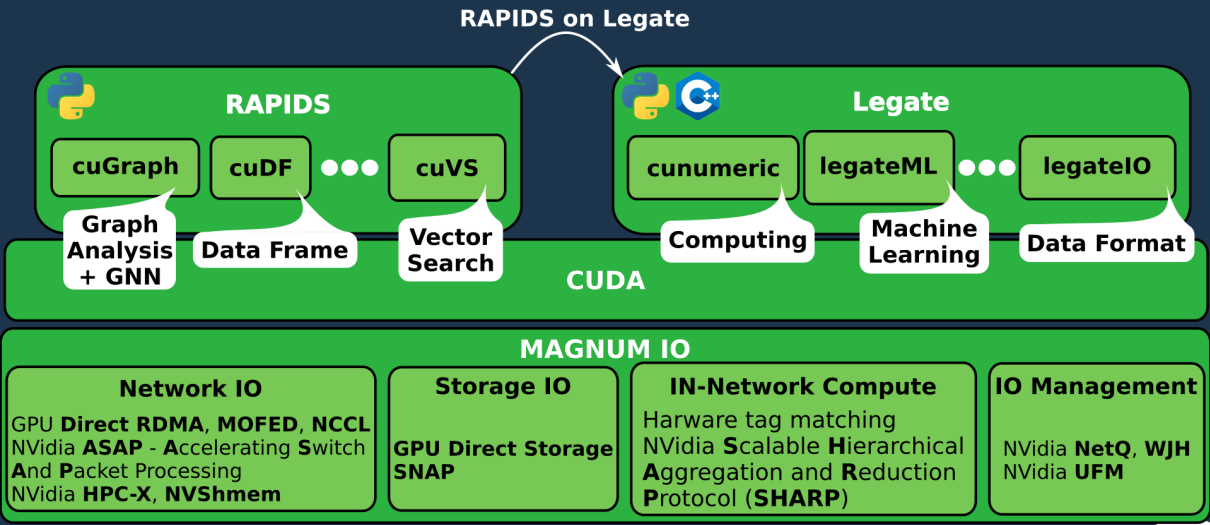
RAPIDS and Legate Frameworks



RAPIDS and Legate Frameworks



RAPIDS and Legate Frameworks



cuDF : Cuda Data Frame

cuDF : Cuda Data Frame

pandas

Core functionnalities but on GPU

cuDF : Cuda Data Frame

pandas

Core functionnalities but on GPU

~ 10 to **100x faster** than Pandas

cuDF : Cuda Data Frame

pandas

Core functionnalities but on GPU

~ 10 to **100x faster** than Pandas

Pandas example

```python
import pandas as pd
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = pd.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

cuDF : Cuda Data Frame

Core functionnalities but on GPU

~ 10 to **100x faster** than Pandas

Pandas example

```python
import pandas as pd
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = pd.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

cuDF equivalent

```python
import cudf
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = cudf.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

cuDF : Cuda Data Frame

pandas

Core functionnalities but on GPU

~ 10 to **100x faster** than Pandas

Pandas example

```
import pandas as pd
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = pd.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

cuDF equivalent

```
import cudf
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = cudf.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

I don't want to change my scripts

cuDF : Cuda Data Frame

pandas

Core functionnalities but on GPU

~ 10 to **100x faster** than Pandas

Pandas example

```python
import pandas as pd
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = pd.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

cuDF equivalent

```python
import cudf
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = cudf.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

I don't want to change my scripts

Fine, so just use :
python -m cudf.pandas script.py

cuDF : Cuda Data Frame

pandas

Core functionnalities but on GPU

~ 10 to **100x faster** than Pandas

Pandas example

```python
import pandas as pd
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = pd.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

cuDF equivalent

```python
import cudf
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = cudf.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

I don't want to change my scripts

Fine, so just use :
python -m cudf.pandas script.py

With **Juyter :**
%load_ext cudf.pandas

cuDF : Cuda Data Frame

Laboratoire d'Annecy de Physique des Particules

pandas

Core functionnalities but on GPU

~ 10 to **100x faster** than Pandas

Pandas example

```python
import pandas as pd
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = pd.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

cuDF equivalent

```python
import cudf
import requests
from io import StringIO

url = "https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode("utf-8")

tips_df = cudf.read_csv(StringIO(content))
tips_df["tip_percentage"] = tips_df["tip"] / tips_df["total_bill"] * 100

# display average tip by dining party size
print(tips_df.groupby("size").tip_percentage.mean())
```

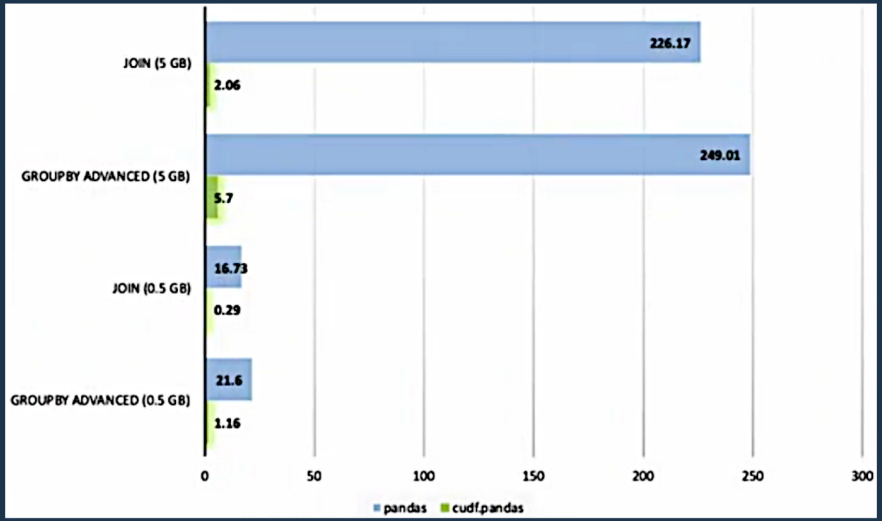
I don't want to change my scripts

cuDF passes **94%** of **Pandas unit tests** (some edge cases have to be solved)

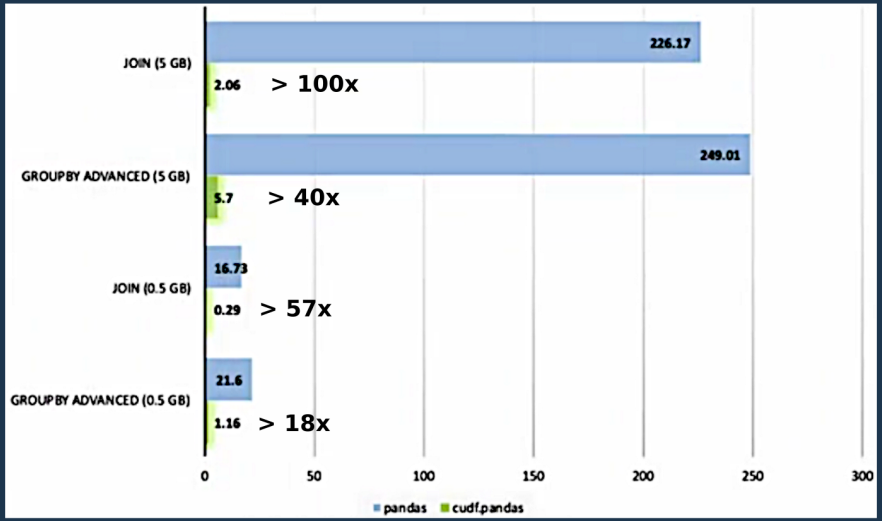
Fine, so just use :
python -m cudf.pandas script.py

With **Juyter :**
%load_ext cudf.pandas

cuDF : Cuda Data Frame



cuDF : Cuda Data Frame



cuDF : Cuda Data Frame

Stay on GPU as much as possible

cuDF : Cuda Data Frame

Stay on GPU as much as possible

Pay **attention** to **GPU memory usage**
(when **GPU** is **FULL**, operations go **back to CPU**,
so this can lead to **unexpected slow down**

cuDF : Cuda Data Frame

Stay on GPU as much as possible

Pay **attention** to **GPU memory usage**
(when **GPU** is **FULL**, operations go **back to CPU**,
so this can lead to **unexpected slow down**

cuDF profiler :
- **from cudf.pandas import Profiler**
- **%cudf.pandas.profiler**

cuDF : Cuda Data Frame

Stay on GPU as much as possible

cuDF profiler :
- **from cudf.pandas import Profiler**
- **%cudf.pandas.profiler**

Pay **attention** to **GPU memory usage**
(when **GPU** is **FULL**, operations go **back to CPU**,
so this can lead to **unexpected slow down**

Use **Idiomatic pandas pattern** as much as possible

not **user defined functions**

cuDF : Cuda Data Frame

Stay on GPU as much as possible

cuDF profiler :
- **from cudf.pandas import Profiler**
- **%cudf.pandas.profiler**

No Dependency on NumPy for now

Maybe official in Pandas 3

Pay **attention** to **GPU memory usage** (when **GPU** is **FULL**, operations go **back to CPU**, so this can lead to **unexpected slow down**

Use **Idiomatic pandas pattern** as much as possible

not **user defined functions**

cuDF : Cuda Data Frame

Stay on GPU as much as possible

cuDF profiler :
- **from cudf.pandas import Profiler**
- **%cudf.pandas.profiler**

No Dependency on NumPy for now

Maybe official in Pandas 3

Pay **attention** to **GPU memory usage** (when **GPU** is **FULL**, operations go **back to CPU**, so this can lead to **unexpected slow down**

Use **Idiomatic pandas pattern** as much as possible

not **user defined functions**

For multiple GPU : dask.cudf

cuVS : Cuda Vector Search

cuVS : Cuda Vector Search

Search in **Vector Database**

cuVS : Cuda Vector Search

Search in **Vector Database**

Vector to : **text**, **video**, **molecules**, etc

cuVS : Cuda Vector Search

Search in **Vector Database**

Vector to : **text**, **video**, **molecules**, etc

Retrieval-Augmented Generation (RAG)
- **Avoid hallucination**
- **Improve accuracy**

cuVS : Cuda Vector Search

Search in **Vector Database**

Vector to : **text**, **video**, **molecules**, etc

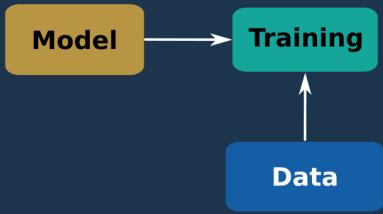
Retrieval-Augmented Generation (RAG)
- **Avoid hallucination**
- **Improve accuracy**

Model

cuVS : Cuda Vector Search

Search in **Vector Database**

Vector to : **text**, **video**, **molecules**, etc



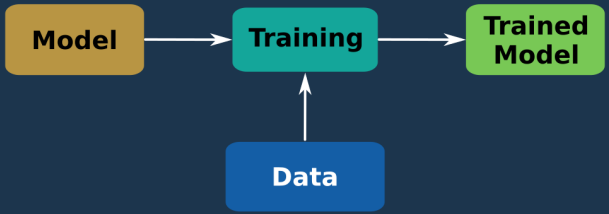
Retrieval-Augmented Generation (RAG)
- **Avoid hallucination**
- **Improve accuracy**

cuVS : Cuda Vector Search

Search in **Vector Database**

Vector to : **text**, **video**, **molecules**, etc

Retrieval-Augmented Generation (RAG)
- **Avoid hallucination**
- **Improve accuracy**

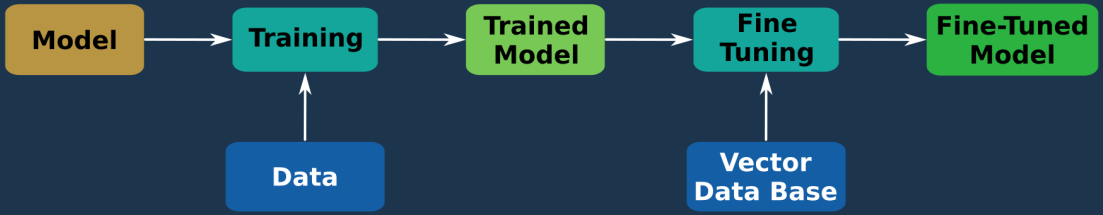


cuVS : Cuda Vector Search

Search in **Vector Database**

Vector to : **text**, **video**, **molecules**, etc

Retrieval-Augmented Generation (RAG)
- **Avoid hallucination**
- **Improve accuracy**

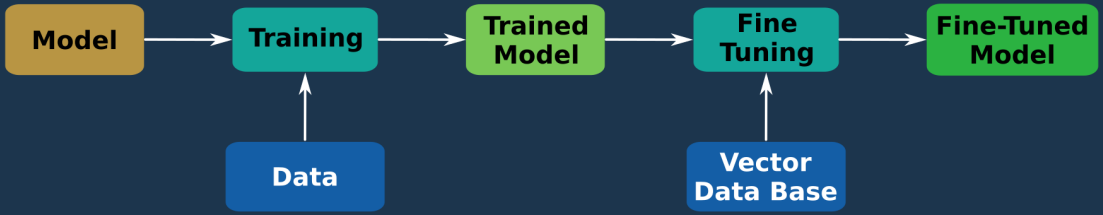


cuVS : Cuda Vector Search

Search in **Vector Database**

Vector to : **text**, **video**, **molecules**, etc

Retrieval-Augmented Generation (RAG)
- **Avoid hallucination**
- **Improve accuracy**



GPU Accelerated

cuGraph : Graph at network scale

Graphs are everywhere : **Images**, **Text**, **Molecules**,
Networks (**Computer**, **Social**, **Citation**), **Product Catalogs**, etc

cuGraph : Graph at network scale

Graphs are everywhere : **Images**, **Text**, **Molecules**,
Networks (**Computer**, **Social**, **Citation**), **Product Catalogs**, etc

Graphs exploit meaning of sparsity

cuGraph : Graph at network scale

Graphs are everywhere : **Images**, **Text**, **Molecules**,
Networks (**Computer**, **Social**, **Citation**), **Product Catalogs**, etc

cuGraph : Graph at network scale

Graphs are everywhere : **Images**, **Text**, **Molecules**,
Networks (**Computer**, **Social**, **Citation**), **Product Catalogs**, etc

cuGraph : Graph at network scale

Graphs are everywhere : **Images**, **Text**, **Molecules**,
Networks (**Computer**, **Social**, **Citation**), **Product Catalogs**, etc

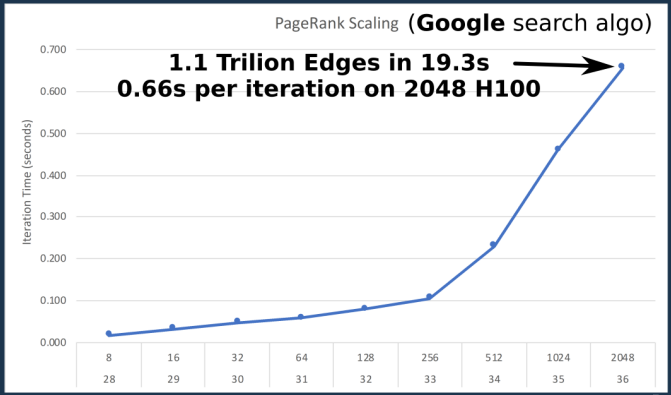
Need **Larger Graphs** (**Network Scale**) :
- **100 TB** Graph
- **1 Trillion** Edges
- Feature Size **~1 k** vector lengths

cuGraph : Graph at network scale

Graphs are everywhere : **Images**, **Text**, **Molecules**,
Networks (**Computer**, **Social**, **Citation**), **Product Catalogs**, etc

Graphs exploit meaning of sparsity → **Sparse Matrix** → **Tensor Cores**

Need **Larger Graphs** (**Network Scale**) :
- **100 TB** Graph
- **1 Trillion** Edges
- Feature Size **~1 k** vector lengths



cuGraph : Libraries

Fast GPU Graph Accelerated Algorithms

Class	Algorithms	MNMG
Centrality	Katz	Yes
	Betweenness Centrality	Yes
	Edge Betweenness Centrality	Yes
	Eigenvector Centrality	Yes
	Degree Centrality (Python only)	Yes
Community	Leiden	Yes
	Louvain	Yes
	Ensemble Clustering for Graphs	Python in 24.04
	Spectral-Clustering - Balanced Cut	
	Spectral-Clustering - Modularity	
	Subgraph Extraction	Yes
	Triangle Counting	Yes
	K-Truss	
Components	Weakly Connected Components	Yes
	Strongly Connected Components	
Core	K-Core	Yes
	Core Number	Yes
Layout	Force Atlas 2	not planned
Link Analysis	PageRank	Yes
	Personal PageRank	Yes
	HITS	Yes
Link Prediction / Similarity	Jaccard Similarity	Yes
	Weighted Jaccard Similarity	Yes
	Overlap Similarity	Yes
	Sorensen	Yes
Traversal	Breadth First Search (BFS)	Yes
	Single Source Shortest Path (SSSP)	Yes
Sampling	Random Walks (Uniform and Biased)	SG Only
	EgoNet	Yes
	Node2Vec	SG Only
	Neighborhood sampling	Yes
Other	Minimum/Maximum Spanning Tree	not planned
	Hungarian	not planned

cuGraph : Libraries

- cuGraph
- pylibcugraph

Fast GPU Graph Accelerated Algorithms

Class	Algorithms	MNMG
Centrality	Katz	Yes
	Betweenness Centrality	Yes
	Edge Betweenness Centrality	Yes
	Eigenvector Centrality	Yes
	Degree Centrality (Python only)	Yes
Community	Leiden	Yes
	Louvain	Yes
	Ensemble Clustering for Graphs	Python in 24.04
	Spectral-Clustering - Balanced Cut	
	Spectral-Clustering - Modularity	
	Subgraph Extraction	Yes
	Triangle Counting	Yes
	K-Truss	
Components	Weakly Connected Components	Yes
	Strongly Connected Components	
Core	K-Core	Yes
	Core Number	Yes
Layout	Force Atlas 2	not planned
Link Analysis	PageRank	Yes
	Personal PageRank	Yes
	HITS	Yes
Link Prediction / Similarity	Jaccard Similarity	Yes
	Weighted Jaccard Similarity	Yes
	Overlap Similarity	Yes
	Sorensen	Yes
Traversal	Breadth First Search (BFS)	Yes
	Single Source Shortest Path (SSSP)	Yes
Sampling	Random Walks (Uniform and Biased)	SG Only
	EgoNet	Yes
	Node2Vec	SG Only
	Neighborhood sampling	Yes
Other	Minimum/Maximum Spanning Tree	not planned
	Hungarian	not planned

cuGraph : Libraries

- cuGraph
- pylibcugraph

Integration :
- **DGL (cugraph-dgl)**
- **PyG (cugraph-pyg)**

Fast GPU Graph Accelerated Algorithms

Class	Algorithms	MNMG
Centrality	Katz	Yes
	Betweenness Centrality	Yes
	Edge Betweenness Centrality	Yes
	Eigenvector Centrality	Yes
	Degree Centrality (Python only)	Yes
Community	Leiden	Yes
	Louvain	Yes
	Ensemble Clustering for Graphs	Python in 24.04
	Spectral-Clustering - Balanced Cut	
	Spectral-Clustering - Modularity	
	Subgraph Extraction	Yes
	Triangle Counting	Yes
	K-Truss	
Components	Weakly Connected Components	Yes
	Strongly Connected Components	
Core	K-Core	Yes
	Core Number	Yes
Layout	Force Atlas 2	not planned
Link Analysis	PageRank	Yes
	Personal PageRank	Yes
	HITS	Yes
Link Prediction / Similarity	Jaccard Similarity	Yes
	Weighted Jaccard Similarity	Yes
	Overlap Similarity	Yes
	Sorensen	Yes
Traversal	Breadth First Search (BFS)	Yes
	Single Source Shortest Path (SSSP)	Yes
Sampling	Random Walks (Uniform and Biased)	SG Only
	EgoNet	Yes
	Node2Vec	SG Only
	Neighborhood sampling	Yes
Other	Minimum/Maximum Spanning Tree	not planned
	Hungarian	not planned

cuGraph : Libraries

- **cuGraph**
- **pylibcugraph**

- **libcugraph**
- **MTMG** (new API coming soon)

Integration :
- **DGL** (**cugraph-dgl**)
- **PyG** (**cugraph-pyg**)

Fast GPU Graph Accelerated Algorithms

Class	Algorithms	MNMG
Centrality	Katz	Yes
	Betweenness Centrality	Yes
	Edge Betweenness Centrality	Yes
	Eigenvector Centrality	Yes
	Degree Centrality (Python only)	Yes
Community	Leiden	Yes
	Louvain	Yes
	Ensemble Clustering for Graphs	Python in 24.04
	Spectral-Clustering - Balanced Cut	
	Spectral-Clustering - Modularity	
	Subgraph Extraction	Yes
	Triangle Counting	Yes
	K-Truss	
Components	Weakly Connected Components	Yes
	Strongly Connected Components	
Core	K-Core	Yes
	Core Number	Yes
Layout	Force Atlas 2	not planned
Link Analysis	PageRank	Yes
	Personal PageRank	Yes
	HITS	Yes
Link Prediction / Similarity	Jaccard Similarity	Yes
	Weighted Jaccard Similarity	Yes
	Overlap Similarity	Yes
	Sorensen	Yes
Traversal	Breadth First Search (BFS)	Yes
	Single Source Shortest Path (SSSP)	Yes
Sampling	Random Walks (Uniform and Biased)	SG Only
	EgoNet	Yes
	Node2Vec	SG Only
	Neighborhood sampling	Yes
Other	Minimum/Maximum Spanning Tree	not planned
	Hungarian	not planned

cuGraph : Libraries

- **cuGraph**
- **pylibcugraph**

Integration :
- **DGL (cugraph-dgl)**
- **PyG (cugraph-pyg)**

- **libcugraph**
- **MTMG** (new API coming soon)

- **libcugraph_c**

Fast GPU Graph Accelerated Algorithms

Class	Algorithms	MNMG
Centrality	Katz	Yes
	Betweenness Centrality	Yes
	Edge Betweenness Centrality	Yes
	Eigenvector Centrality	Yes
	Degree Centrality (Python only)	Yes
Community	Leiden	Yes
	Louvain	Yes
	Ensemble Clustering for Graphs	Python in 24.04
	Spectral-Clustering - Balanced Cut	
	Spectral-Clustering - Modularity	
	Subgraph Extraction	Yes
	Triangle Counting	Yes
	K-Truss	
Components	Weakly Connected Components	Yes
	Strongly Connected Components	
Core	K-Core	Yes
	Core Number	Yes
Layout	Force Atlas 2	not planned
Link Analysis	PageRank	Yes
	Personal PageRank	Yes
	HITS	Yes
Link Prediction / Similarity	Jaccard Similarity	Yes
	Weighted Jaccard Similarity	Yes
	Overlap Similarity	Yes
	Sorensen	Yes
Traversal	Breadth First Search (BFS)	Yes
	Single Source Shortest Path (SSSP)	Yes
Sampling	Random Walks (Uniform and Biased)	SG Only
	EgoNet	Yes
	Node2Vec	SG Only
	Neighborhood sampling	Yes
Other	Minimum/Maximum Spanning Tree	not planned
	Hungarian	not planned

cuGraph : Libraries

- cuGraph
- pylibcugraph

Integration :
- **DGL** (**cugraph-dgl**)
- **PyG** (**cugraph-pyg**)

- **libcugraph**
- **MTMG** (new API coming soon)

C
- **libcugraph_c**

Service
- **cuGraph as Service** (coming soon)

Fast GPU Graph Accelerated Algorithms

Class	Algorithms	MNMG
Centrality	Katz	Yes
	Betweenness Centrality	Yes
	Edge Betweenness Centrality	Yes
	Eigenvector Centrality	Yes
	Degree Centrality (Python only)	Yes
Community	Leiden	Yes
	Louvain	Yes
	Ensemble Clustering for Graphs	Python in 24.04
	Spectral-Clustering - Balanced Cut	
	Spectral-Clustering - Modularity	
	Subgraph Extraction	Yes
	Triangle Counting	Yes
	K-Truss	
Components	Weakly Connected Components	Yes
	Strongly Connected Components	
Core	K-Core	Yes
	Core Number	Yes
Layout	Force Atlas 2	not planned
Link Analysis	PageRank	Yes
	Personal PageRank	Yes
	HITS	Yes
Link Prediction / Similarity	Jaccard Similarity	Yes
	Weighted Jaccard Similarity	Yes
	Overlap Similarity	Yes
	Sorensen	Yes
Traversal	Breadth First Search (BFS)	Yes
	Single Source Shortest Path (SSSP)	Yes
Sampling	Random Walks (Uniform and Biased)	SG Only
	EgoNet	Yes
	Node2Vec	SG Only
	Neighborhood sampling	Yes
Other	Minimum/Maximum Spanning Tree	not planned
	Hungarian	not planned

NetworkX

Graphs are everywhere (again)

NetworkX

Graphs are everywhere (again)

Open Source project about Graphs :
- **Connectivity**
- **Shortest path**
- **Route planning**
- ...

NetworkX

Graphs are everywhere (again)

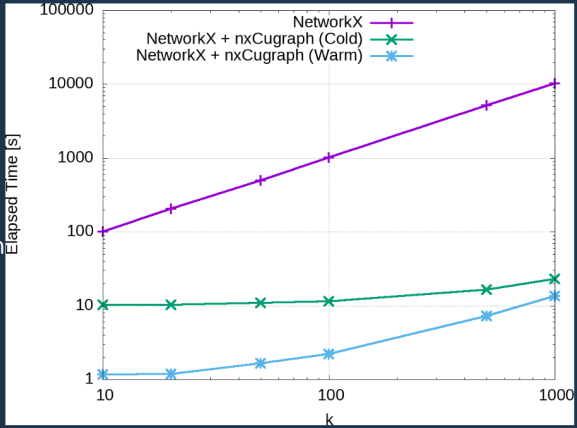
Open Source project about Graphs :
- **Connectivity**
- **Shortest path**
- **Route planning**
- ...

No need to **change** script

NETWORK_BACKEND_PRIORITY=**cugraph python** bc_demo.py

NetworkX

- **Dataset** : U.S. patent dataset, National Bureau of Economic Research (https://snap.stanford.edu/data.cit-Patents.html)
- **Directed Graph** : **3.7 M Nodes**, **16.5 M Edges**
- **CPU** : Intel Xeon Gold 6128 CPU @ 3.40 GHz, 45 GB
- **GPU** : NVidia Quadro RTX 8000, 48 GB



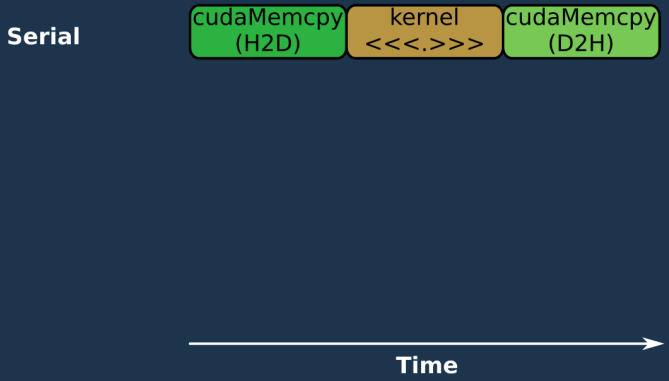
NETWORKX_BACKEND_PRIORITY="cugraph" python

```
>>> import networkx as nx
>>> G = nx.DiGraph()
...
>>> nx.betweenness_centrality(G, k=k)
```

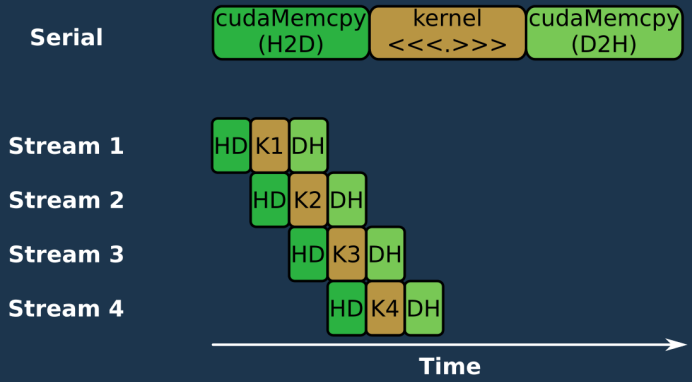
https://github.com/rapidsai/cugraph/blob/branch-24.04/benchmarks/nx-cugraph/pytest-based/bench_algos.py

RMM : Rapids Memory Management

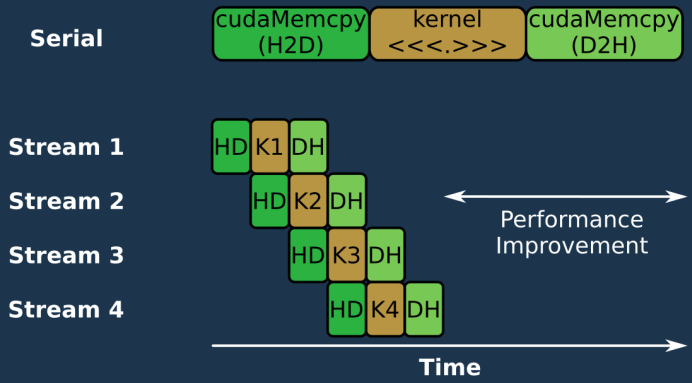
RMM : Rapids Memory Management



RMM : Rapids Memory Management

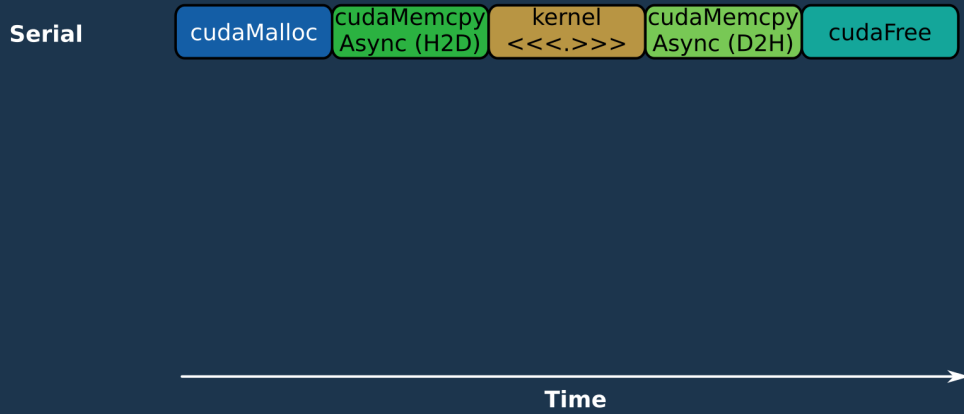


RMM : Rapids Memory Management

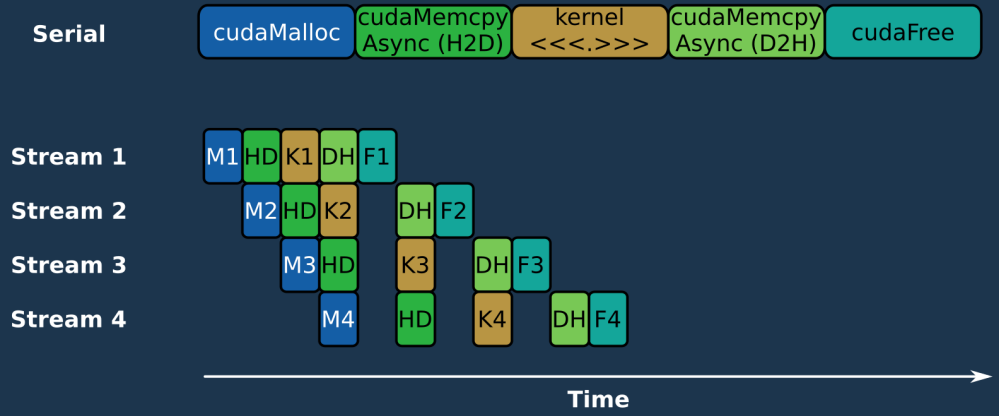


RMM : Rapids Memory Management

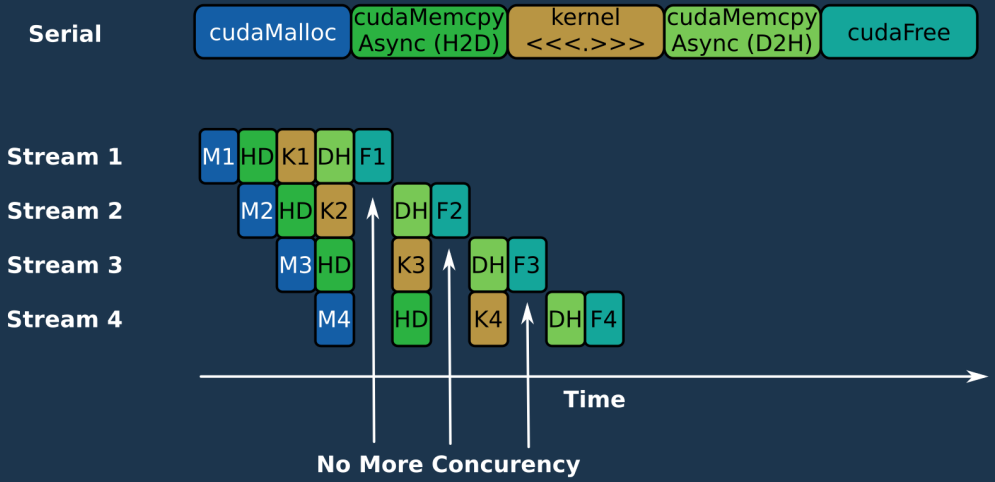
RMM : Rapids Memory Management



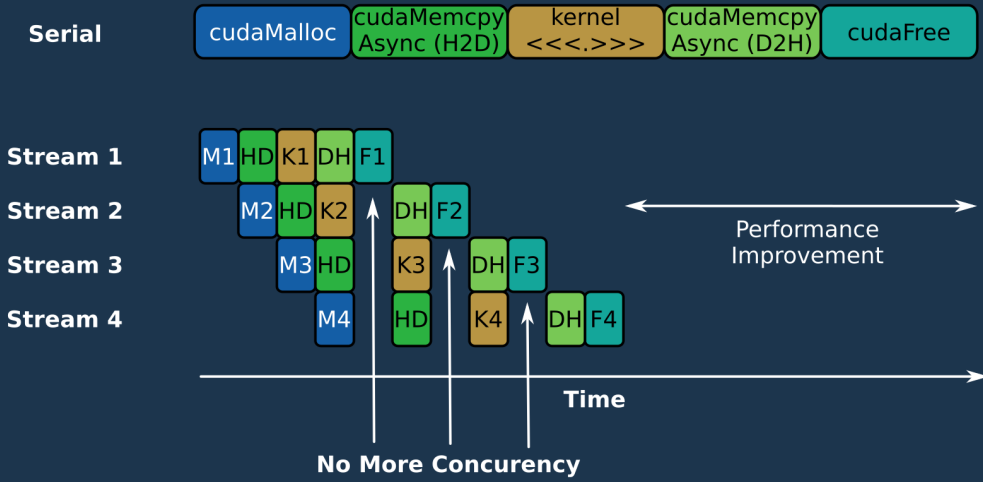
RMM : Rapids Memory Management



RMM : Rapids Memory Management

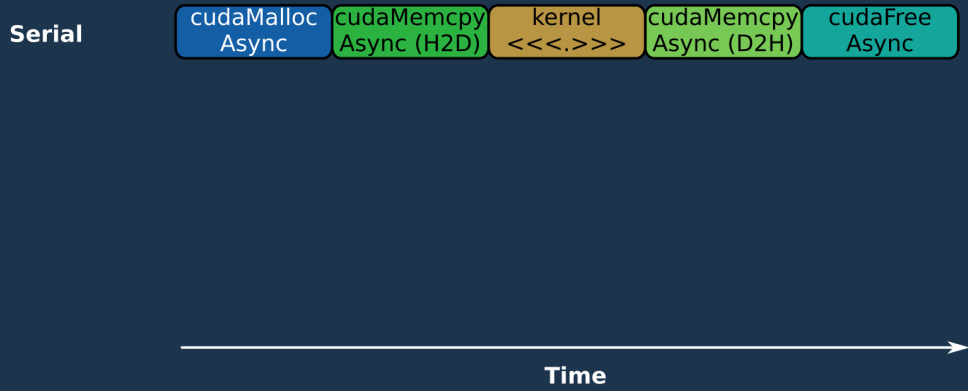


RMM : Rapids Memory Management

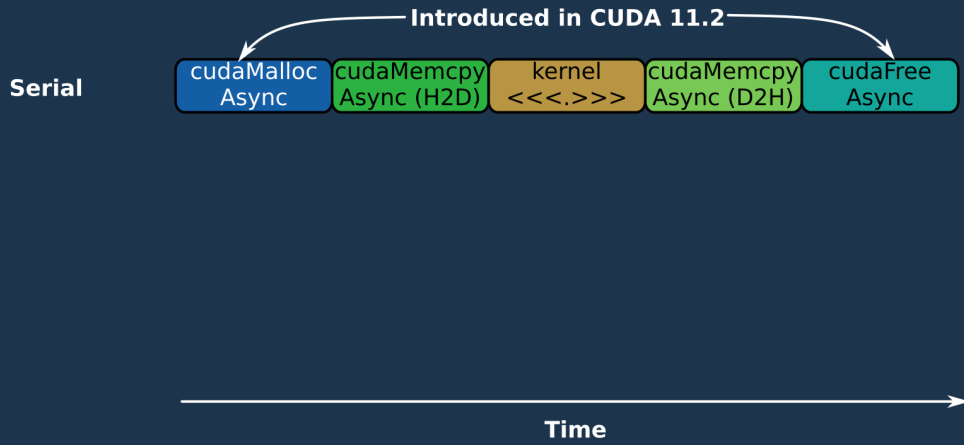


RMM : Rapids Memory Management

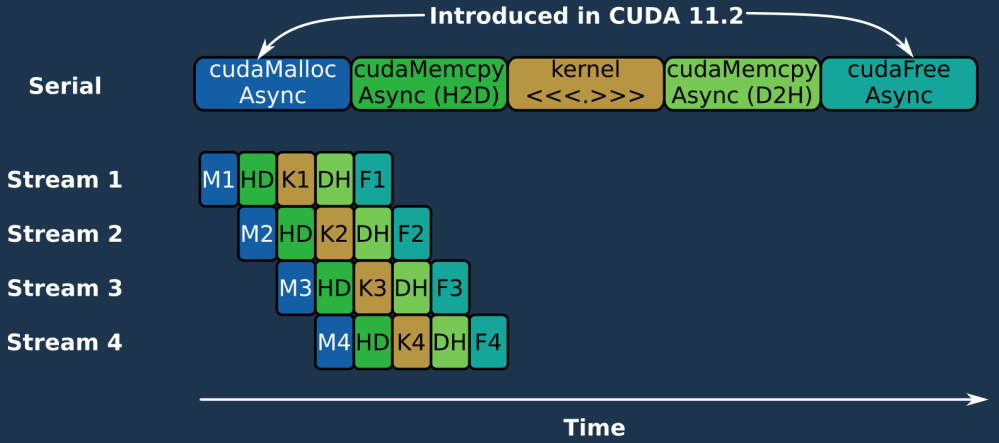
RMM : Rapids Memory Management



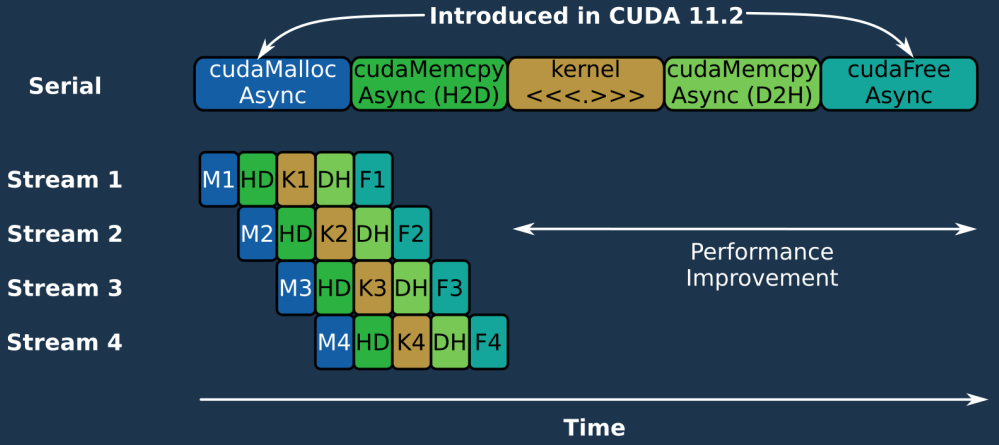
RMM : Rapids Memory Management



RMM : Rapids Memory Management



RMM : Rapids Memory Management



RMM : Rapids Memory Management

RMM : Rapids Memory Management

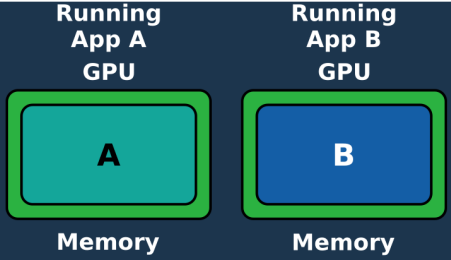
Running
App A

GPU

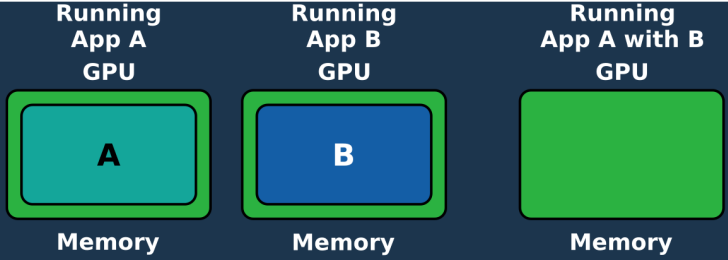
Memory

RMM : Rapids Memory Management

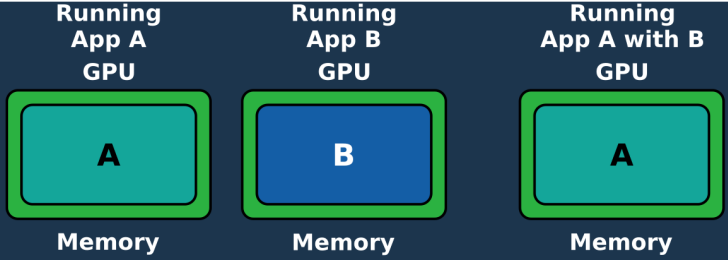
RMM : Rapids Memory Management



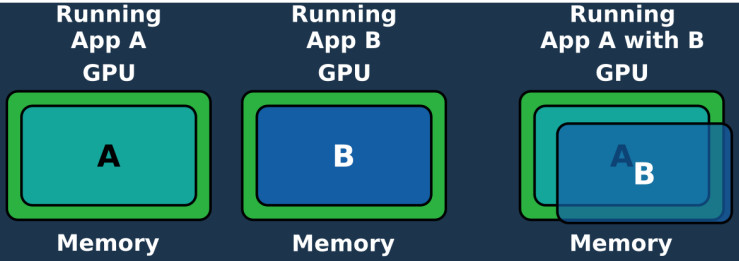
RMM : Rapids Memory Management



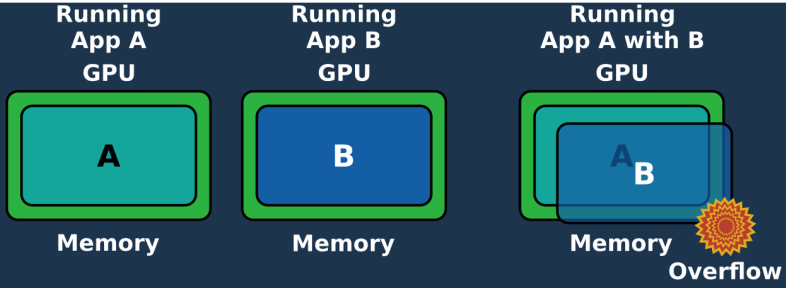
RMM : Rapids Memory Management



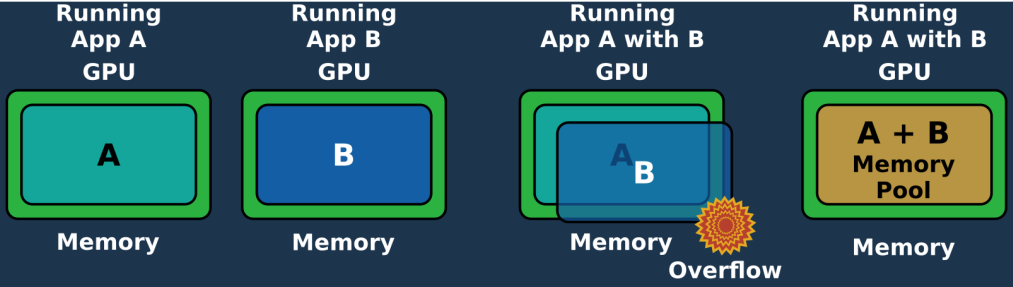
RMM : Rapids Memory Management



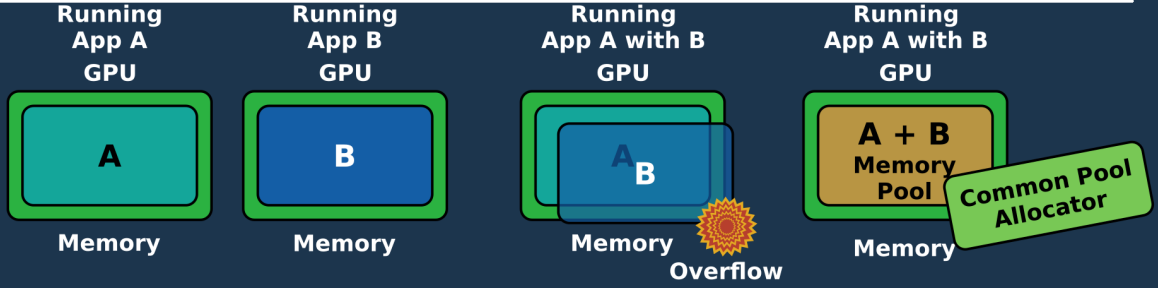
RMM : Rapids Memory Management



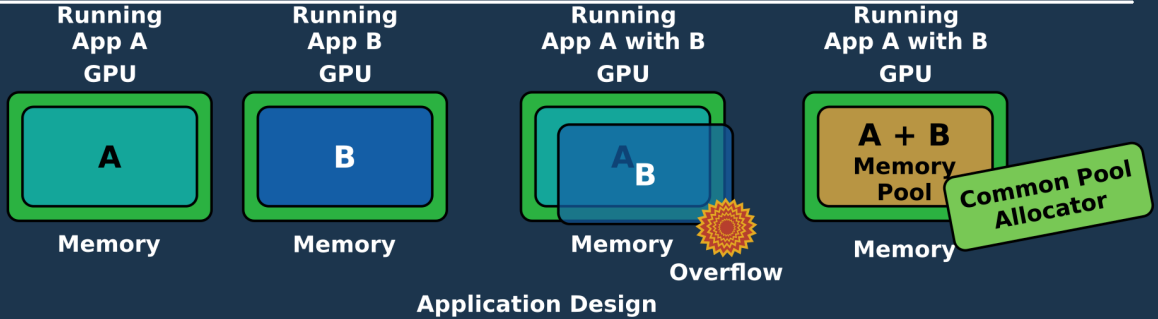
RMM : Rapids Memory Management



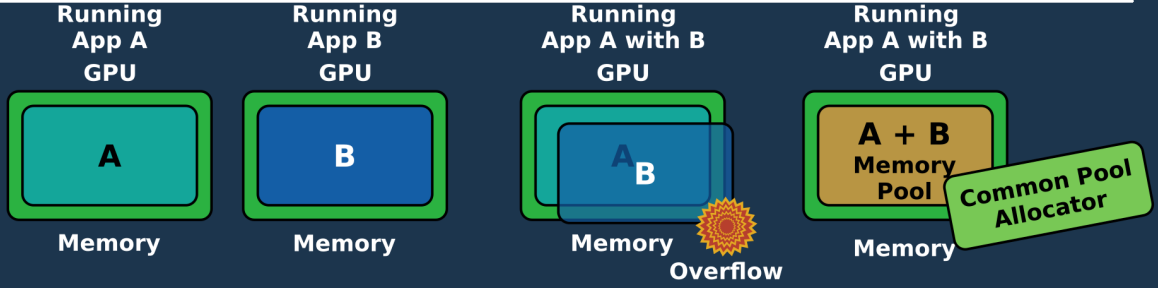
RMM : Rapids Memory Management



RMM : Rapids Memory Management

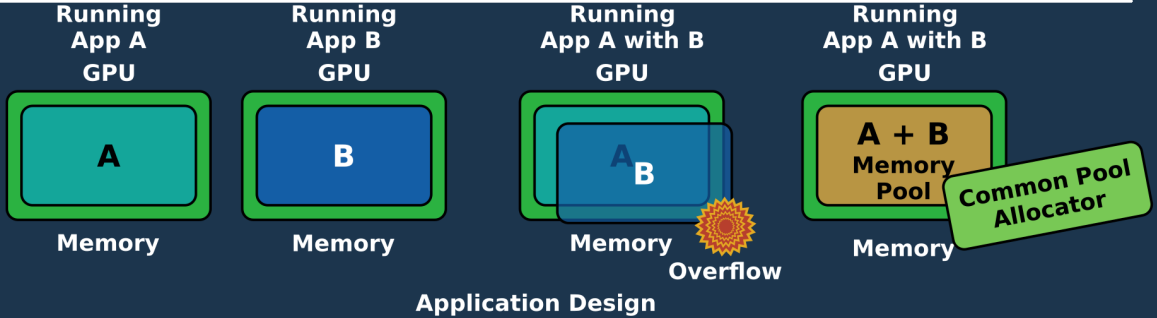


RMM : Rapids Memory Management

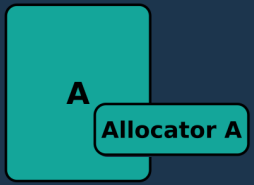


Application Design

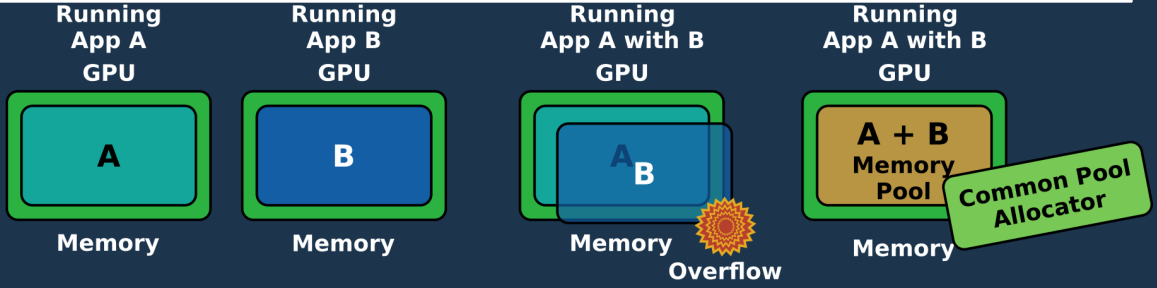
RMM : Rapids Memory Management



Application Design

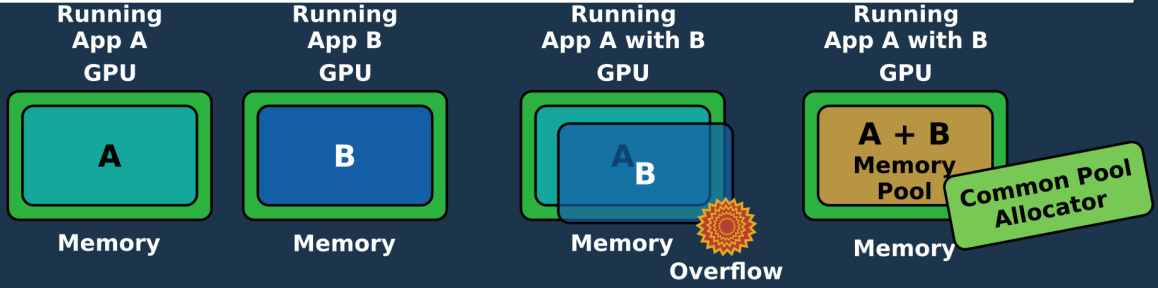


RMM : Rapids Memory Management

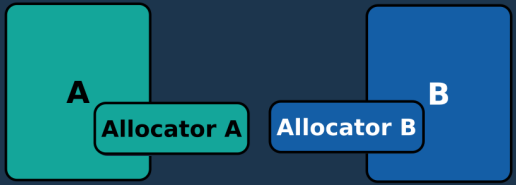


Application Design

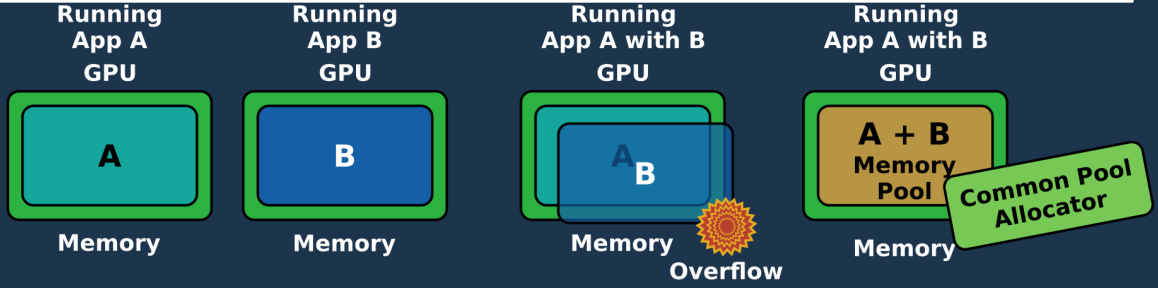
RMM : Rapids Memory Management



Application Design



RMM : Rapids Memory Management



Application Design

RMM : Rapids Memory Management

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

RMM : Rapids Memory Management

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

Common Interface for
Host/device
memory allocation

RMM : Rapids Memory Management

Laboratoire d'Annecy de Physique des Particules

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

Common Interface for
Host/device
memory allocation

cuda::mr::memory_ressource
cuda::mr::async_memory_ressource

RMM : Rapids Memory Management

Laboratoire d'Annecy de Physique des Particules

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

Common Interface for
Host/device
memory allocation

cuda::mr::memory_ressource
cuda::mr::async_memory_ressource

cuda::mr concepts added to **libcudac++**
https://nvidia.github.io/cccl/libcudacxx/

RMM : Rapids Memory Management

Laboratoire d'Annecy de Physique des Particules

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

Common Interface for
Host/device
memory allocation

cuda::mr::memory_ressource
cuda::mr::async_memory_ressource

cuda::mr concepts added to **libcudac++**
https://nvidia.github.io/cccl/libcudacxx/

RMM : Rapids Memory Management

Laboratoire d'Annecy de Physique des Particules

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

Common Interface for
Host/device
memory allocation

Custom
Allocator

cuda::mr::memory_ressource
cuda::mr::async_memory_ressource

cuda::mr concepts added to **libcudac++**
https://nvidia.github.io/cccl/libcudacxx/

RMM : Rapids Memory Management

Laboratoire d'Annecy de Physique des Particules

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

Common Interface for
Host/device
memory allocation

Essential for libraries cooperation

Custom Allocator

cuda::mr::memory_ressource
cuda::mr::async_memory_ressource

cuda::mr concepts added to **libcudac++**
https://nvidia.github.io/cccl/libcudacxx/

RMM : Rapids Memory Management

Laboratoire d'Annecy de Physique des Particules

Fast and Flexible Memory Management

Customize Allocation :
- **Pool**, **Cached**, **Fixed Size**
- **Logging**, **Reporting**, **Statistics**, **Debugging**
- **Tracking**, **Leak Detection**
- **Encryption**
- **Padding**, **Alignment**
- **Prototyping**
- etc

https://github.com/rapidsai/rmm/

Common Interface for
Host/device
memory allocation

Essential for libraries cooperation

Custom Allocator

cuda::mr::memory_ressource
cuda::mr::async_memory_ressource

Used in :
- **cuDF**
- **cuML**
- **RAFT**

cuda::mr concepts added to **libcudac++**

https://nvidia.github.io/cccl/libcudacxx/

Legate

Legate

Accelerate Computing of well used **Python packages**


Legate

Accelerate Computing of well used **Python packages**

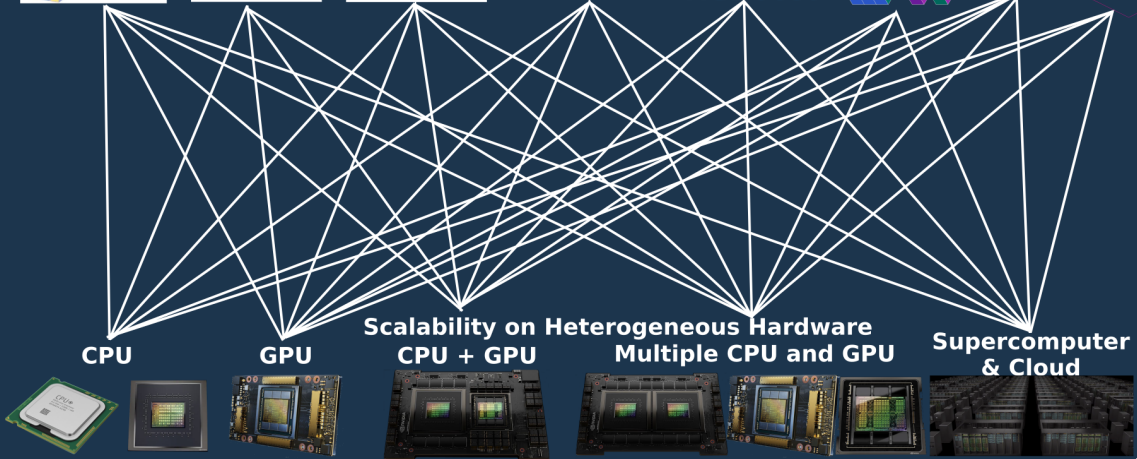
NumPy SciPy pandas scikit learn dmlc XGBoost HDF

Scalability on Heterogeneous Hardware

CPU **GPU** **CPU + GPU** **Multiple CPU and GPU** **Supercomputer & Cloud**

Legate

Accelerate Computing of well used **Python packages**



Legate

Accelerate Computing of well used **Python packages**

Too much work
Impossible to maintain

Scalability on Heterogeneous Hardware

CPU | **GPU** | **CPU + GPU** | **Multiple CPU and GPU** | **Supercomputer & Cloud**

Legate

Accelerate Computing of well used **Python packages**

NumPy SciPy pandas scikit learn dmlc XGBoost CuPy HDF

Legate - Productivity and Composable layer

Legion - Implicit Parallelism Layer

Realm - Runtime for Scalable and Portable Execution

Scalability on Heterogeneous Hardware

CPU **GPU** **CPU + GPU** **Multiple CPU and GPU** **Supercomputer & Cloud**

Legate

Accelerate Computing of well used **Python packages**

NumPy | SciPy | pandas | scikit learn | dmlc XGBoost | JAX | HDF

cuNumeric | **Legate DataFrame** | **Legate ML** | **Legate Boost** | **Legate JAX** | **Legate IO**

Legate - Productivity and Composable layer

Legion - Implicit Parallelism Layer

Realm - Runtime for Scalable and Portable Execution

Scalability on Heterogeneous Hardware

CPU | **GPU** | **CPU + GPU** | **Multiple CPU and GPU** | **Supercomputer & Cloud**

Legate

Accelerate Computing of well used **Python packages**

NumPy | SciPy | pandas | scikit learn | dmlc XGBoost | HDF

cuNumeric | **Legate DataFrame** | **Legate ML** | **Legate Boost** | **Legate JAX** | **Legate IO**

Legate - Productivity and Composable layer

Legion - Implicit Parallelism Layer

Realm - Runtime for Scalable and Portable Execution

Coherent Backends

Coherent Transfers & Allocations

Scalability on Heterogeneous Hardware

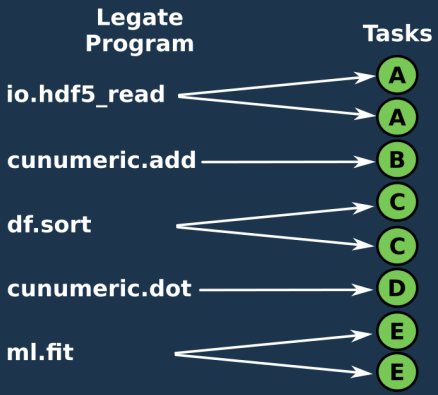
CPU | GPU | CPU + GPU | Multiple CPU and GPU | Supercomputer & Cloud

Legate Scaling

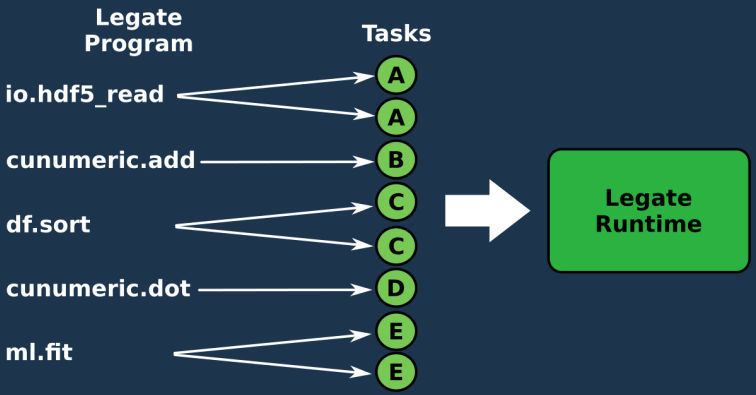
Legate Scaling

Legate Program
io.hdf5_read
cunumeric.add
df.sort
cunumeric.dot
ml.fit

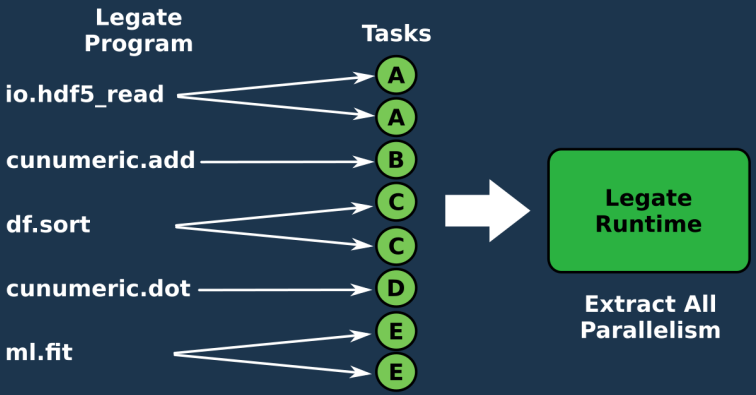
Legate Scaling



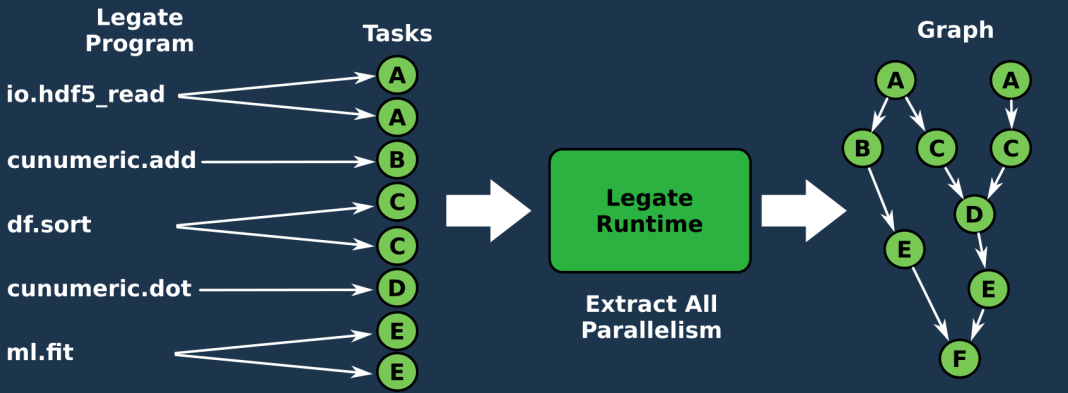
Legate Scaling



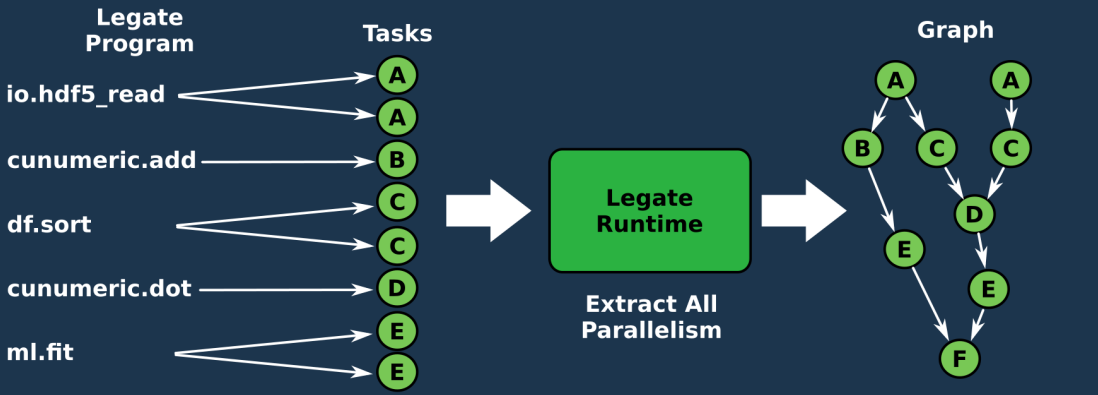
Legate Scaling



Legate Scaling

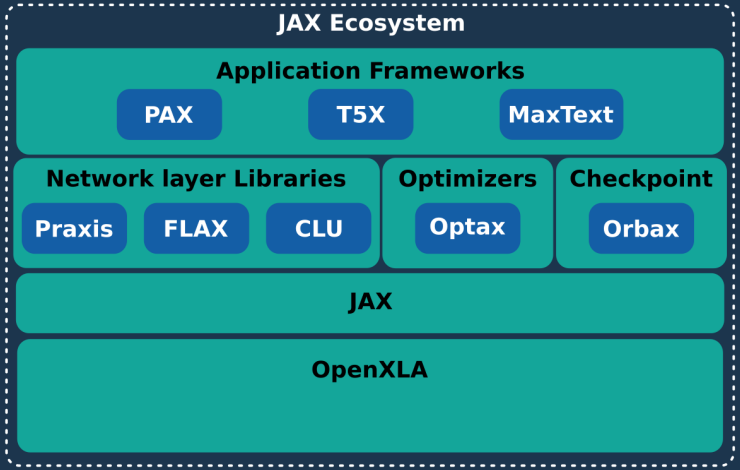


Legate Scaling

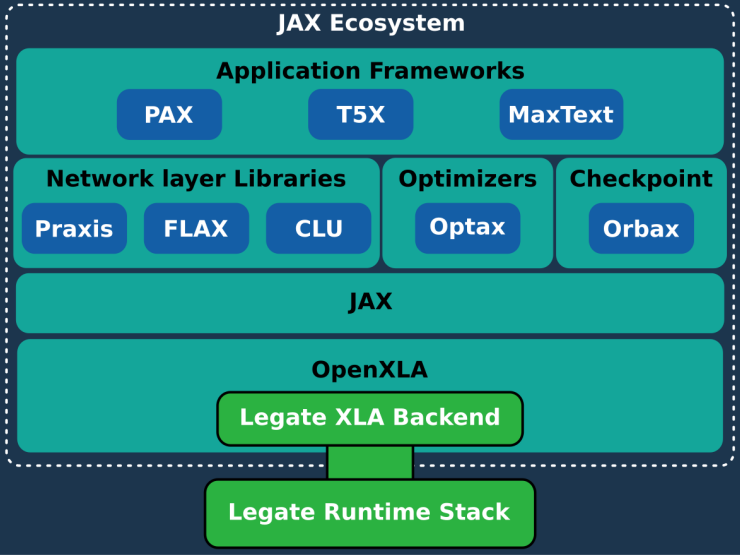


Legate JAX

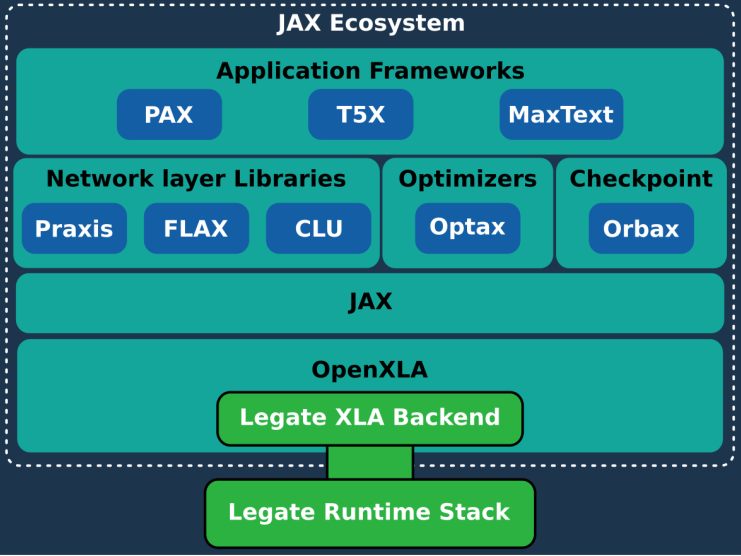
Legate JAX


JAX Ecosystem
Application Frameworks
PAX
T5X
MaxText
Network layer Libraries
Praxis
FLAX
CLU
Optimizers
Optax
Checkpoint
Orbax
JAX
OpenXLA

Legate JAX

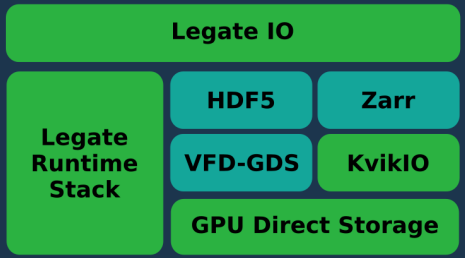

JAX Ecosystem
Application Frameworks
PAX
T5X
MaxText
Network layer Libraries
Praxis
FLAX
CLU
Optimizers
Optax
Checkpoint
Orbax
JAX
OpenXLA
Legate XLA Backend
Legate Runtime Stack

Legate JAX

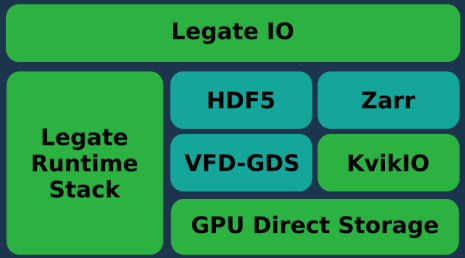


Scalability comparable to State of the art **PAXML (PP)** and **ALPA** on OPT 175B with BFloat16 From **128** to **512 GPU A100**

Legate I/O

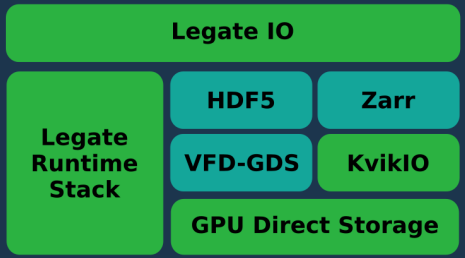

Legate IO
Legate Runtime Stack
HDF5
Zarr
VFD-GDS
KvikIO
GPU Direct Storage

Legate I/O



Real Open-Source Standards

Legate I/O

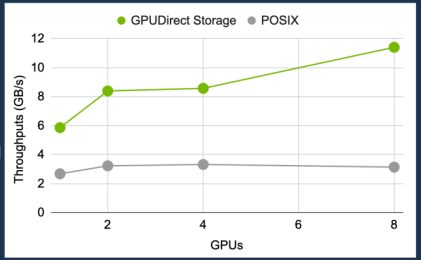


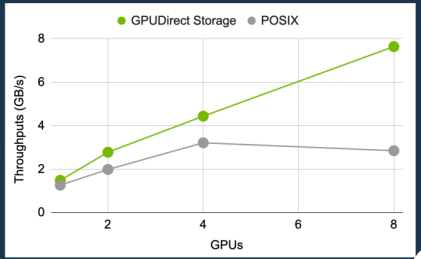
Zarr read with Legate tasking 4GB data per GPU

Real Open-Source Standards

HDF5 read with Legate Tasking 3GB data per GPU

Preliminary weak Scaling Results





Legate CPU/GPU Switch

Example : **multi-physics solver**

```
u, v, p, c, u_coarse, v_coarse, mask = \
    init(nx, ny, nx_coarse, ny_coarse)
for _ in range(nsteps):
    # Run solve with a fine mesh on GPUs

    u, v, p = solve_fluid(u, v, p)
    u_coarse = u[mask].reshape(ny_coarse, nx_coarse)
    v_coarse = v[mask].reshape(ny_coarse, nx_coarse)
    # Run solve with a coarse mesh on CPUs

    c = solve_scalar(c, u_coarse, v_coarse)
```

Legate CPU/GPU Switch

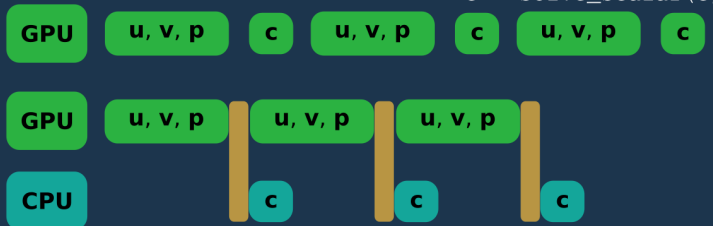
Example : **multi-physics solver**

```
u, v, p, c, u_coarse, v_coarse, mask = \
    init(nx, ny, nx_coarse, ny_coarse)
for _ in range(nsteps):
    # Run solve with a fine mesh on GPUs

    u, v, p = solve_fluid(u, v, p)
    u_coarse = u[mask].reshape(ny_coarse, nx_coarse)
    v_coarse = v[mask].reshape(ny_coarse, nx_coarse)
    # Run solve with a coarse mesh on CPUs

    c = solve_scalar(c, u_coarse, v_coarse)
```

Legate CPU/GPU Switch

Example : **multi-physics solver**

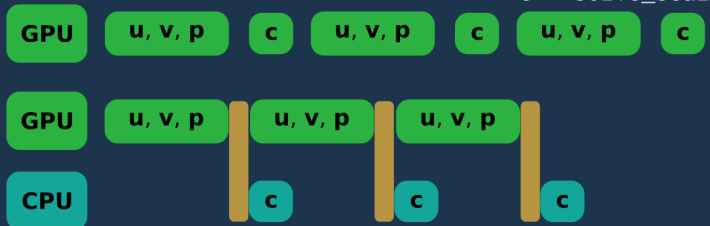
```
u, v, p, c, u_coarse, v_coarse, mask = \
    init(nx, ny, nx_coarse, ny_coarse)
for _ in range(nsteps):
    # Run solve with a fine mesh on GPUs

    u, v, p = solve_fluid(u, v, p)
    u_coarse = u[mask].reshape(ny_coarse, nx_coarse)
    v_coarse = v[mask].reshape(ny_coarse, nx_coarse)
    # Run solve with a coarse mesh on CPUs

    c = solve_scalar(c, u_coarse, v_coarse)
```



Legate CPU/GPU Switch

Example : **multi-physics solver**

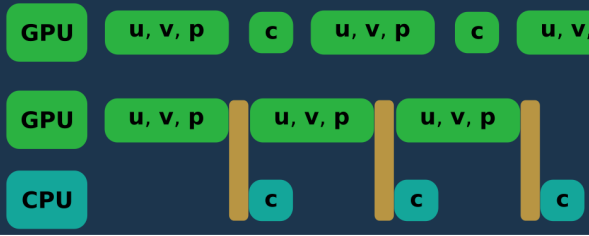
```
from legate.core import get_machine, ProcessorKind
u, v, p, c, u_coarse, v_coarse, mask = \
    init(nx, ny, nx_coarse, ny_coarse)
for _ in range(nsteps):
    # Run solve with a fine mesh on GPUs
    with get_machine().only(ProcessorKind.GPU):
        u, v, p = solve_fluid(u, v, p)
        u_coarse = u[mask].reshape(ny_coarse, nx_coarse)
        v_coarse = v[mask].reshape(ny_coarse, nx_coarse)
    # Run solve with a coarse mesh on CPUs
    with get_machine().only(ProcessorKind.OMP):
        c = solve_scalar(c, u_coarse, v_coarse)
```

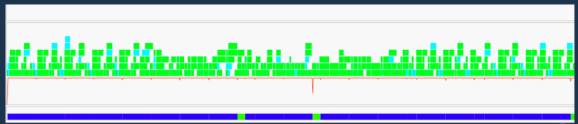


Legate CPU/GPU Switch

Example : **multi-physics solver**

```
from legate.core import get_machine, ProcessorKind
u, v, p, c, u_coarse, v_coarse, mask = \
    init(nx, ny, nx_coarse, ny_coarse)
for _ in range(nsteps):
    # Run solve with a fine mesh on GPUs
    with get_machine().only(ProcessorKind.GPU):
        u, v, p = solve_fluid(u, v, p)
        u_coarse = u[mask].reshape(ny_coarse, nx_coarse)
        v_coarse = v[mask].reshape(ny_coarse, nx_coarse)
    # Run solve with a coarse mesh on CPUs
    with get_machine().only(ProcessorKind.OMP):
        c = solve_scalar(c, u_coarse, v_coarse)
```





Legate Kernel Fusion

Legate Kernel Fusion

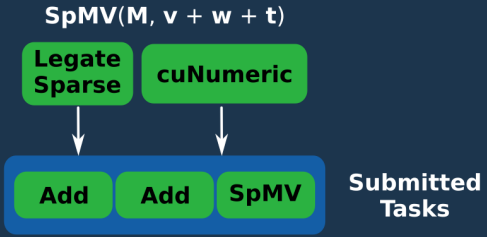
SpMV(**M**, **v** + **w** + **t**)

Legate Kernel Fusion

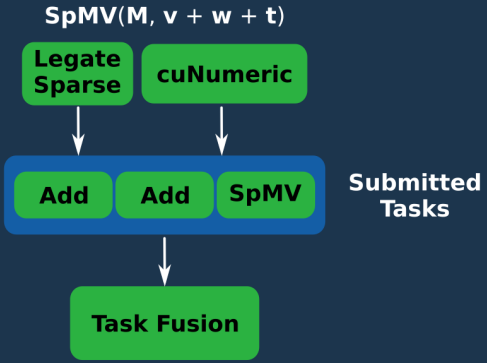
SpMV(**M**, **v** + **w** + **t**)

Legate Sparse **cuNumeric**

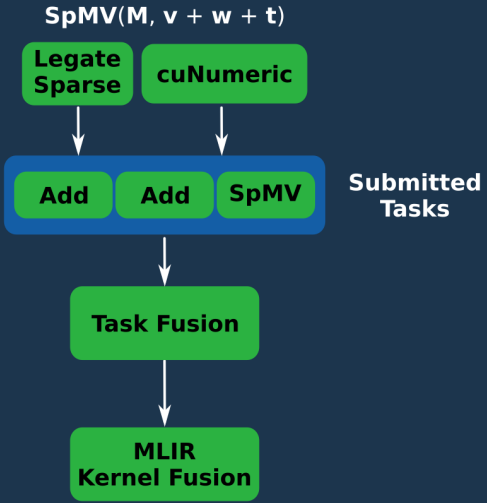
Legate Kernel Fusion



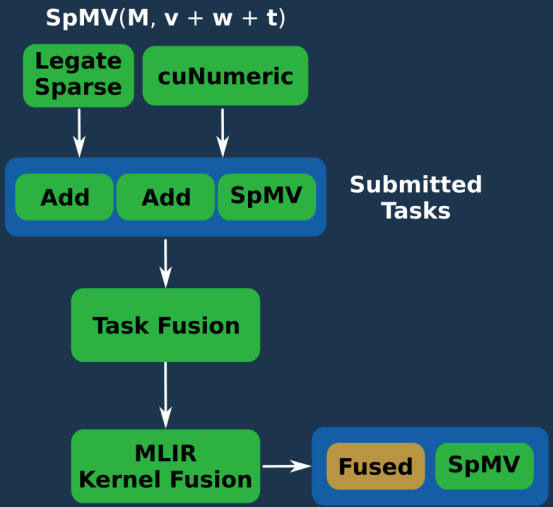
Legate Kernel Fusion



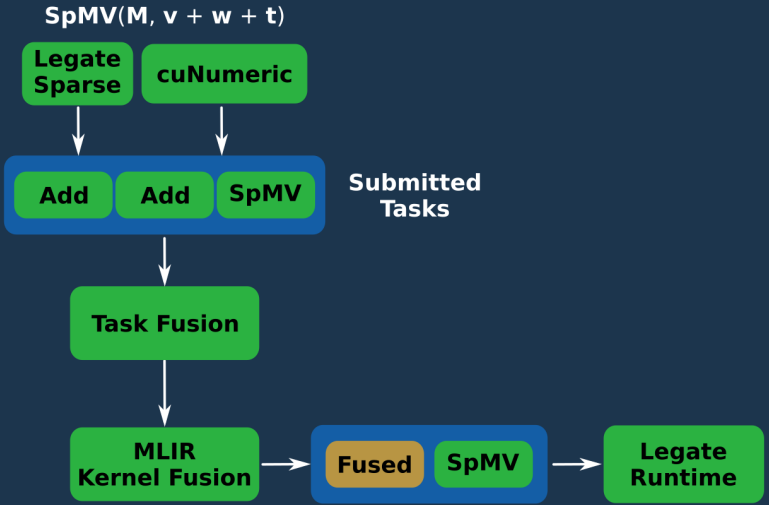
Legate Kernel Fusion



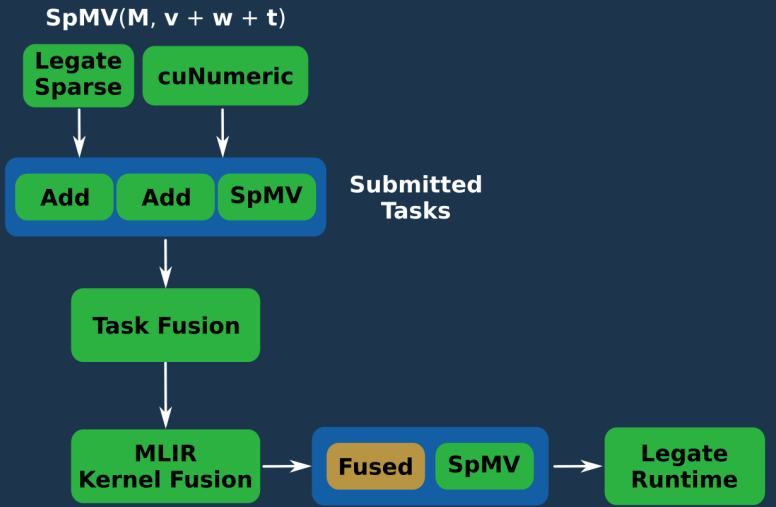
Legate Kernel Fusion



Legate Kernel Fusion



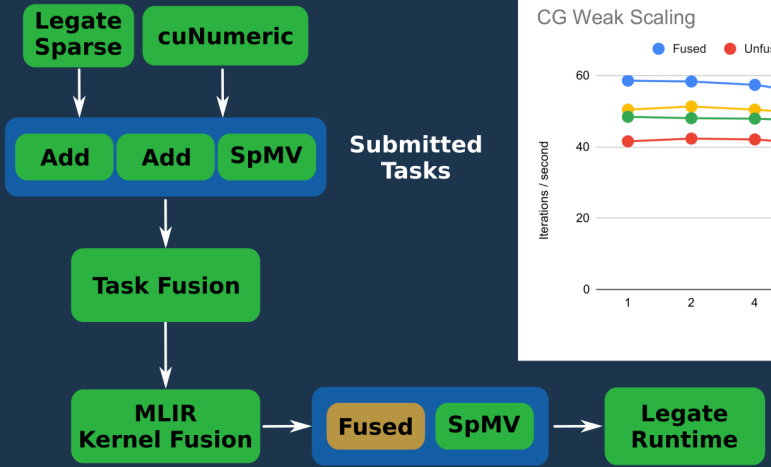
Legate Kernel Fusion



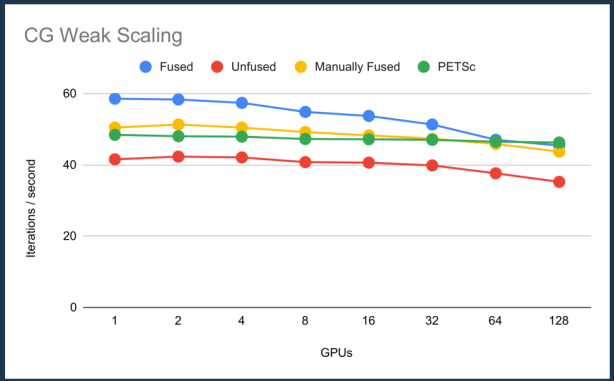
- **No tmp allocation**
- **2** kernel **launch** instead of **3**
- **No Synchro** between kernels
- **No Data Movement**

Legate Kernel Fusion

SpMV(M, v + w + t)



Conjugate Gradient Throughput on DGX-A100 cluster



- **No tmp allocation**
- **2** kernel **launch** instead of **3**
- **No Synchro** between kernels
- **No Data Movement**

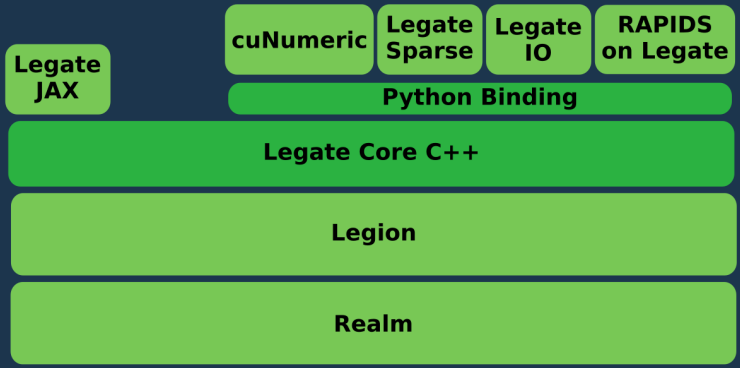
Legate Core

Legate Core

Complete **rewrite** of the **core** framework in **C++**

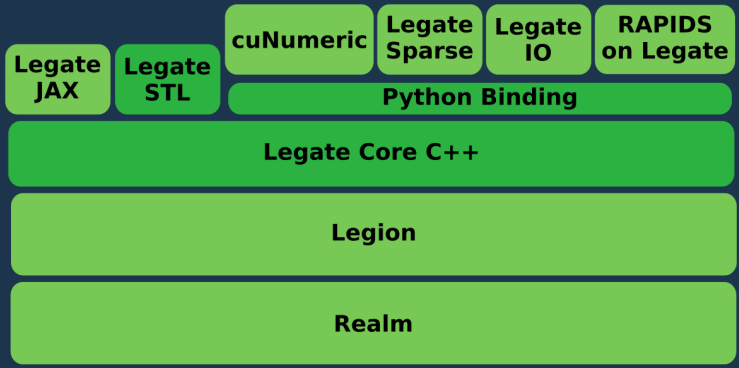
Legate Core

Complete **rewrite** of the **core** framework in **C++**



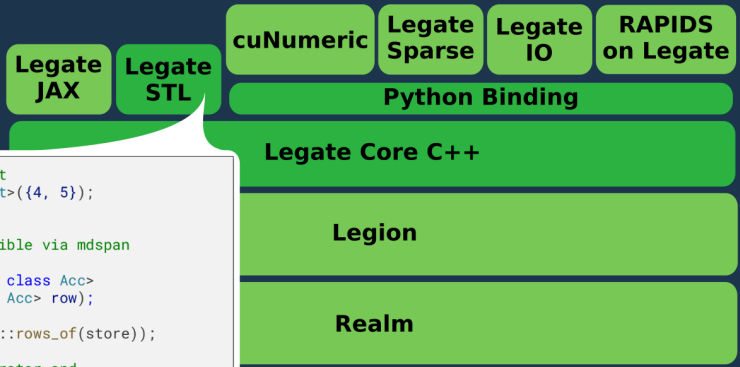
Legate Core

Complete **rewrite** of the **core** framework in **C++**



Legate Core

Complete **rewrite** of the **core** framework in **C++**

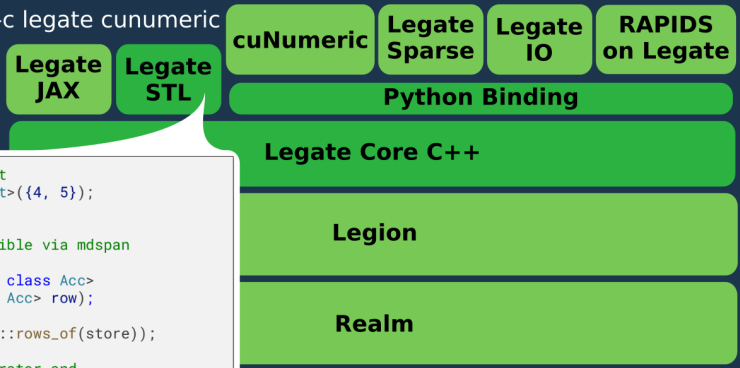



```python
// Create a 2-D Legate store and initialize it
auto store = legate::stl::create_store<int64_t>({4, 5});
legate::stl::fill(store1, 1.);

// Operate on a row of data at a time, accessible via mdspan
struct MungeRow {
  template <class Elem, class Ext, class Map, class Acc>
  void operator()(std::mdspan<Elem, Ext, Map, Acc> row);
};
legate::stl::for_each(MungeRow{}, legate::stl::rows_of(store));

// Transform the store with a pre-defined operator and
// reduce the result with a custom operator, in a single call
auto result =
  legate::stl::transform_reduce(store,
                                legate::stl::scalar{std::int64_t{0}},
                                std::plus{},
                                [](auto x) const { return x * x; });
```

Legate Core

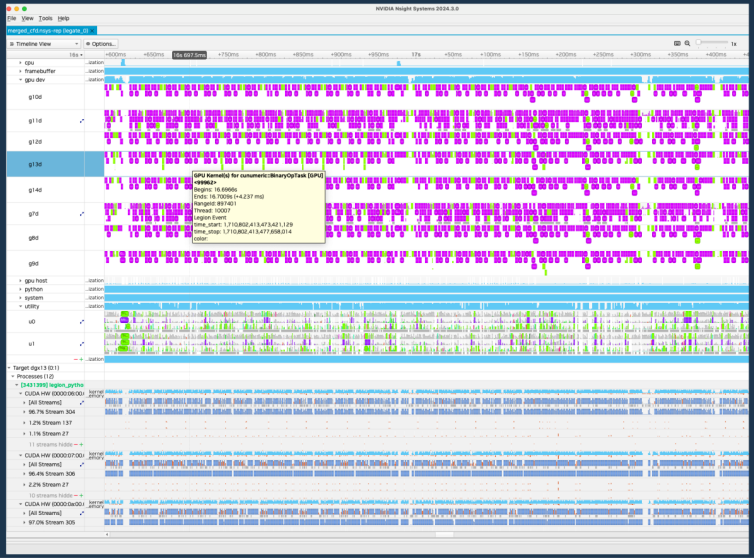
Complete **rewrite** of the **core** framework in **C++**

conda install -c nvidia -c conda-forge -c legate cunumeric

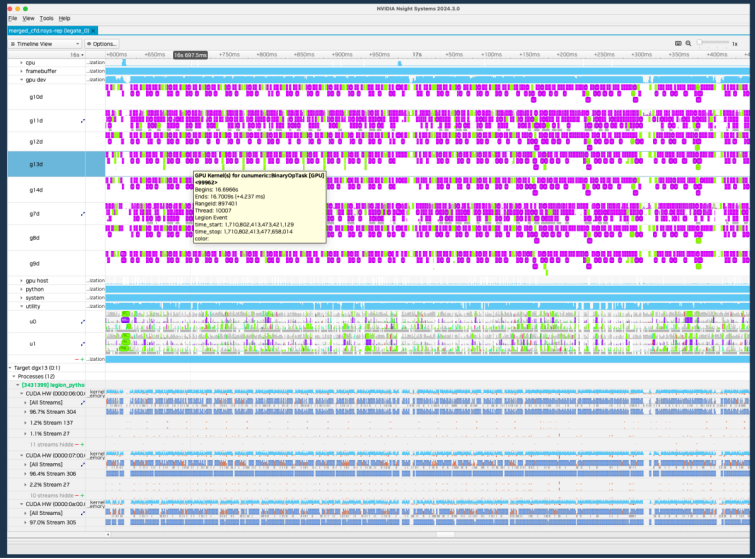



```
// Create a 2-D Legate store and initialize it
auto store = legate::stl::create_store<int64_t>({4, 5});
legate::stl::fill(store1, 1.);

// Operate on a row of data at a time, accessible via mdspan
struct MungeRow {
  template <class Elem, class Ext, class Map, class Acc>
  void operator()(std::mdspan<Elem, Ext, Map, Acc> row);
};
legate::stl::for_each(MungeRow{}, legate::stl::rows_of(store));

// Transform the store with a pre-defined operator and
// reduce the result with a custom operator, in a single call
auto result =
  legate::stl::transform_reduce(store,
                                legate::stl::scalar{std::int64_t{0}},
                                std::plus{},
                                [](auto x) const { return x * x; });
```

Legate and NSight

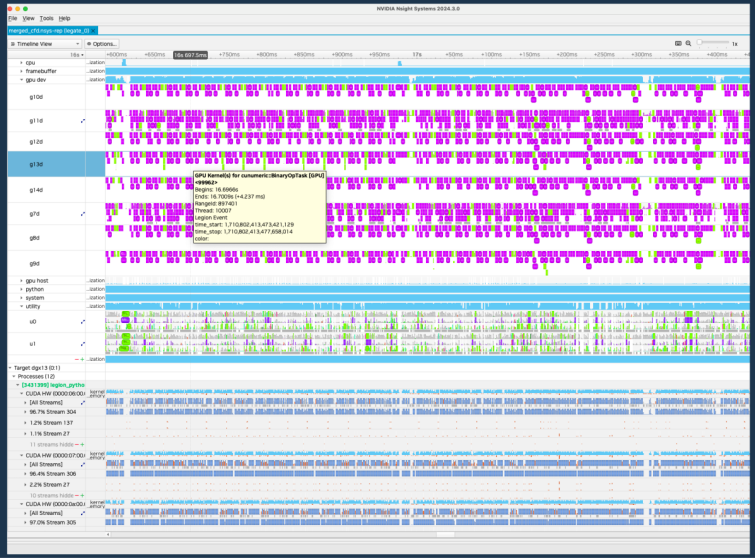


Legate and NSight



Logical view
Legate tasks,
Legate stores,
runtime activities

Legate and NSight



Logical view
Legate tasks,
Legate stores,
runtime activities

Physical view
GPU (kernels), CPU,
OS, network

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Simulation :
- **Rigid Bodies**
- **Particles**
- **Constraints**
- **Geometry**
- **Forces**
- **etc**

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Simulation :
- **Rigid Bodies**
- **Particles**
- **Constraints**
- **Geometry**
- **Forces**
- **etc**

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Core
- Differentiable kernel
- Math (geometry, vector)

Simulation :
- **Rigid Bodies**
- **Particles**
- **Constraints**
- **Geometry**
- **Forces**
- **etc**

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Core
- Differentiable kernel
- Math (geometry, vector)

Sim
- Rigid bodies
- Soft bodies
- Particles
- Cloth
- URDF
- MJCF
- UsdPhysics parsers

Simulation :
- **Rigid Bodies**
- **Particles**
- **Constraints**
- **Geometry**
- **Forces**
- **etc**

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Core
- Differentiable kernel
- Math (geometry, vector)

Sim
- Rigid bodies
- Soft bodies
- Particles
- Cloth
- URDF
- MJCF
- UsdPhysics parsers

Fem
- Differentiable PDE
- Heat transfer
- Diffusion
- Elasticity

Simulation :
- **Rigid Bodies**
- **Particles**
- **Constraints**
- **Geometry**
- **Forces**
- **etc**

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Core
- Differentiable kernel
- Math (geometry, vector)

Sim
- Rigid bodies
- Soft bodies
- Particles
- Cloth
- URDF
- MJCF
- UsdPhysics parsers

Fem
- Differentiable PDE
- Heat transfer
- Diffusion
- Elasticity

LLM
- Code generation

Simulation :
- **Rigid Bodies**
- **Particles**
- **Constraints**
- **Geometry**
- **Forces**
- **etc**

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Core
- Differentiable kernel
- Math (geometry, vector)

Sim
- Rigid bodies
- Soft bodies
- Particles
- Cloth
- URDF
- MJCF
- UsdPhysics parsers

Fem
- Differentiable PDE
- Heat transfer
- Diffusion
- Elasticity

LLM
- Code generation

Simulation :
- **Rigid Bodies**
- **Particles**
- **Constraints**
- **Geometry**
- **Forces**
- **etc**

```python
import warp as wp

@wp.kernel
def integrate(p: wp.array(dtype=wp.vec3),
              v: wp.array(dtype=wp.vec3),
              f: wp.array(dtype=wp.vec3),
              m: wp.array(dtype=float)):

    # thread id
    tid = wp.tid()

    # Semi-implicit Euler step
    v[tid] = v[tid] + (f[tid] * m[tid] + wp.vec3(0.0, -9.8, 0.0)) * dt
    x[tid] = x[tid] + v[tid] * dt

# kernel launch
wp.launch(integrate, dim=1024, inputs=[x, v, f, …], device="cuda:0")
```

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Core
- Differentiable kernel
- Math (geometry, vector)

Sim
- Rigid bodies
- Soft bodies
- Particles
- Cloth
- URDF
- MJCF
- UsdPhysics parsers

Fem
- Differentiable PDE
- Heat transfer
- Diffusion
- Elasticity

LLM
- Code generation

Simulation :
- Rigid Bodies
- Particles
- Constraints
- Geometry
- Forces
- etc

```python
import warp as wp

@wp.kernel
def integrate(p: wp.array(dtype=wp.vec3),
              v: wp.array(dtype=wp.vec3),
              f: wp.array(dtype=wp.vec3),
              m: wp.array(dtype=float)):

    # thread id
    tid = wp.tid()

    # Semi-implicit Euler step
    v[tid] = v[tid] + (f[tid] * m[tid] + wp.vec3(0.0, -9.8, 0.0)) * dt
    x[tid] = x[tid] + v[tid] * dt

# kernel launch
wp.launch(integrate, dim=1024, inputs=[x, v, f, …], device="cuda:0")
```

Interoperability

PyTorch · JAX · NumPy · NanoVDB

Warp : Differentiable Kernels

Differentiable Spatial Computing for Python

Warp

Core
- Differentiable kernel
- Math (geometry, vector)

Sim
- Rigid bodies
- Soft bodies
- Particles
- Cloth
- URDF
- MJCF
- UsdPhysics parsers

Fem
- Differentiable PDE
- Heat transfer
- Diffusion
- Elasticity

LLM
- Code generation

Simulation :
- Rigid Bodies
- Particles
- Constraints
- Geometry
- Forces
- etc

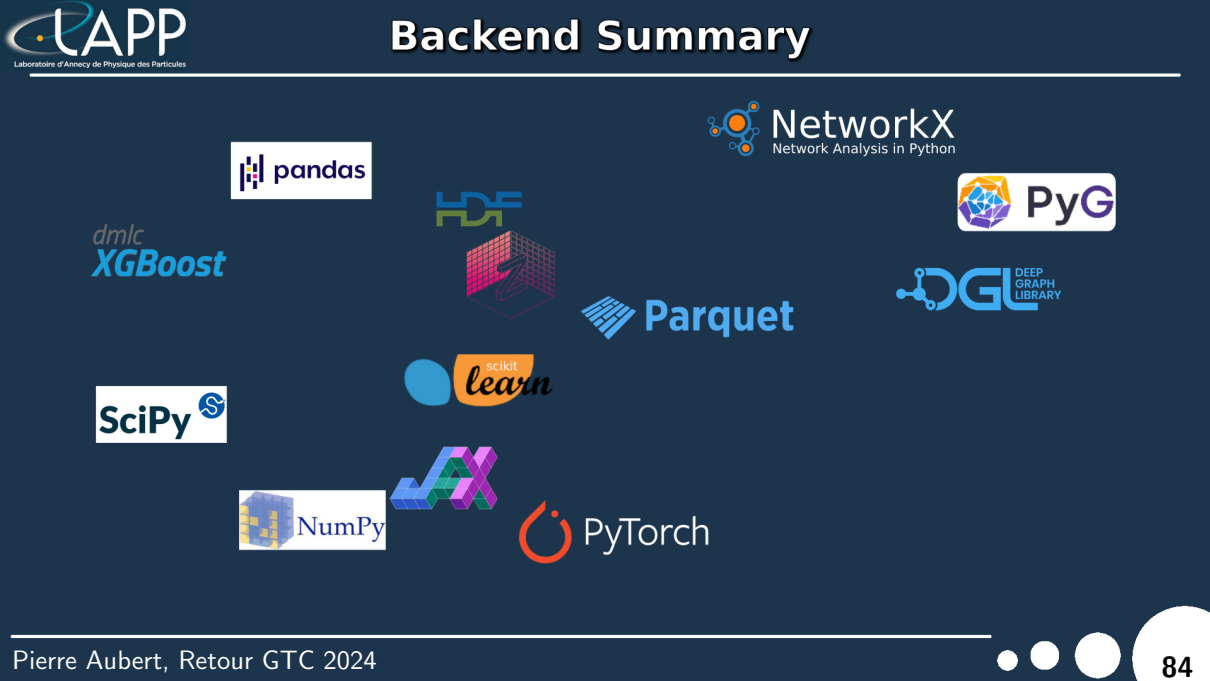
```python
import warp as wp

@wp.kernel
def integrate(p: wp.array(dtype=wp.vec3),
              v: wp.array(dtype=wp.vec3),
              f: wp.array(dtype=wp.vec3),
              m: wp.array(dtype=float)):

    # thread id
    tid = wp.tid()

    # Semi-implicit Euler step
    v[tid] = v[tid] + (f[tid] * m[tid] + wp.vec3(0.0, -9.8, 0.0)) * dt
    x[tid] = x[tid] + v[tid] * dt

# kernel launch
wp.launch(integrate, dim=1024, inputs=[x, v, f, …], device="cuda:0")
```

Interoperability

PyTorch · JAX · NumPy · NanoVDB

pip install warp-lang

Backend Summary



Backend Summary

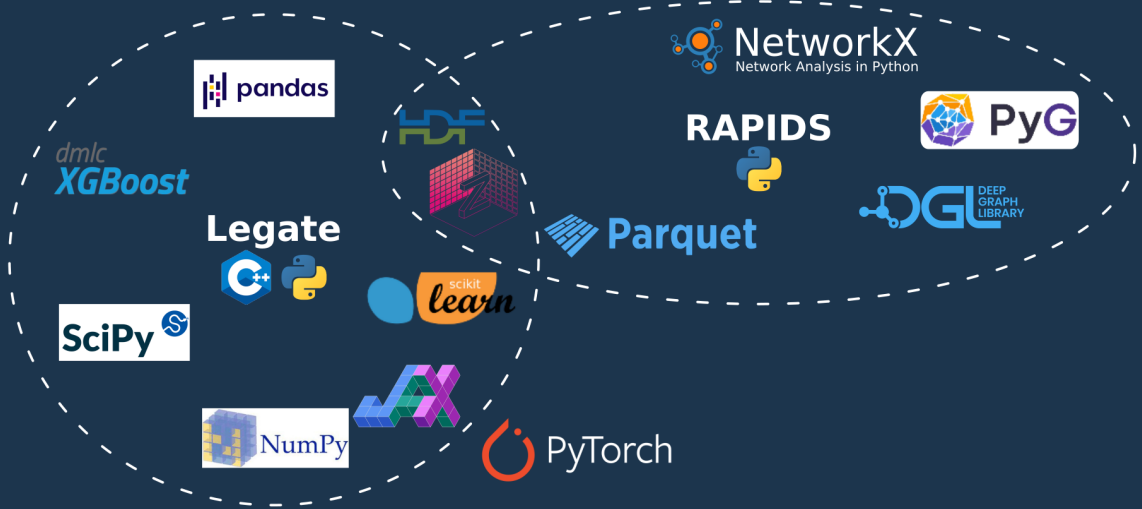
NetworkX
Network Analysis in Python

PyG

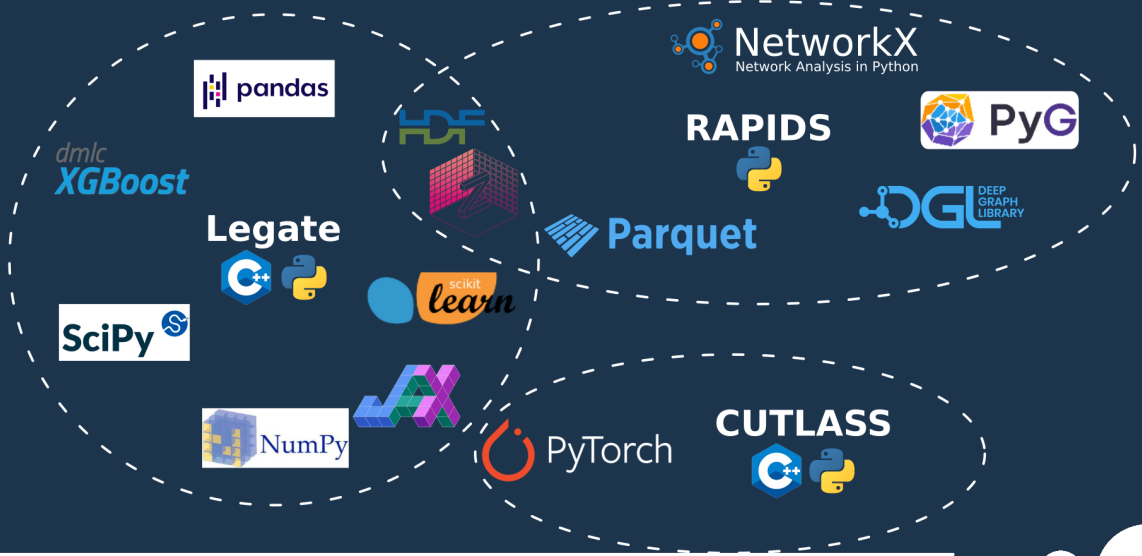
DGL DEEP GRAPH LIBRARY

Parquet

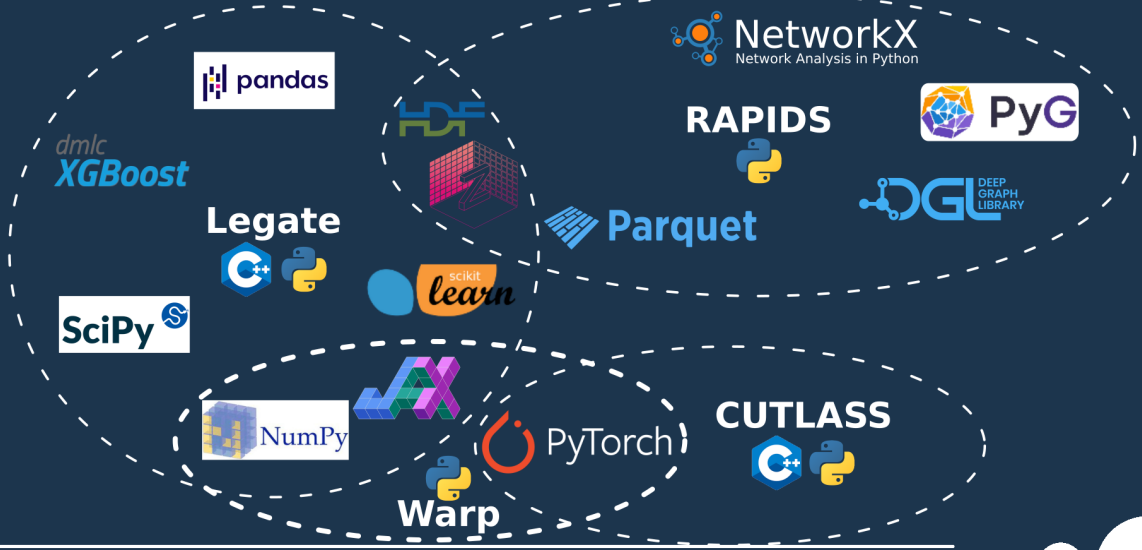
Backend Summary



Backend Summary

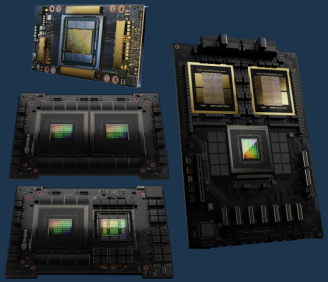


Backend Summary



Chip Design and Simulation

Chip Design and Simulation

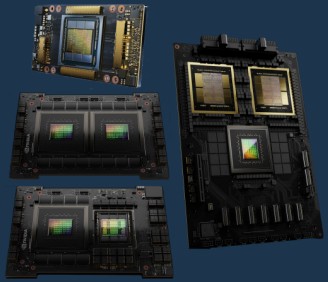


Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Design and Simulation



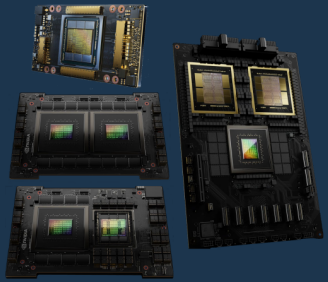
Chip Simulation

Complex Chips

16 months to produce a chip

3 months to print it for real tests

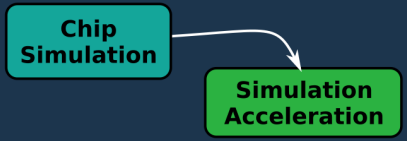
Chip Design and Simulation



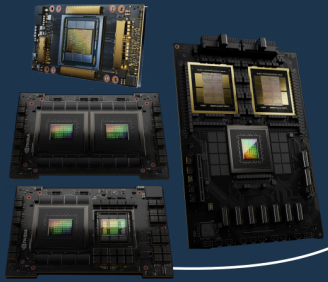
Complex Chips

16 months to produce a chip

3 months to print it for real tests



Chip Design and Simulation



Chip Simulation

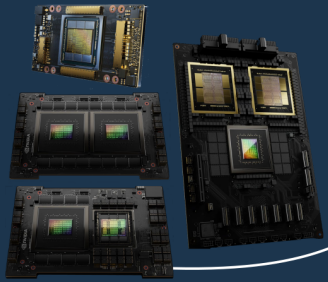
Simulation Acceleration

Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Design and Simulation



Chip Simulation

Simulation Acceleration

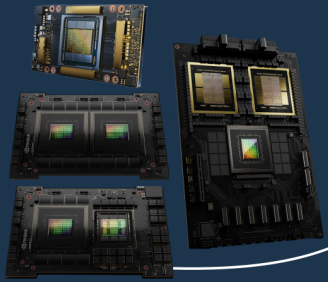
More Accurate Testing

Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Design and Simulation



Chip Simulation

Simulation Acceleration

More Accurate Testing

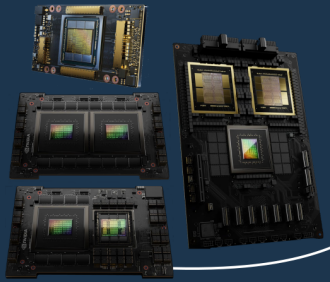
More bugs found with simulation

Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Design and Simulation



Chip Simulation

Simulation Acceleration

More Accurate Testing

More bugs found with simulation

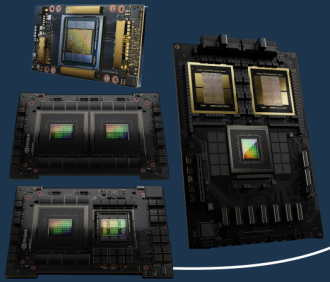
Huge Error Logs

Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Design and Simulation



Chip Simulation

Simulation Acceleration

More Accurate Testing

More bugs found with simulation

Hard to Interpret

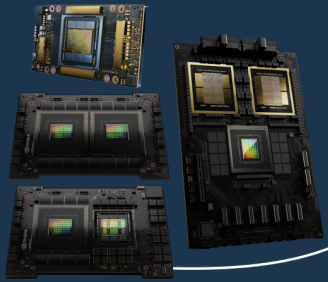
Huge Error Logs

Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Design and Simulation



Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Simulation

Simulation Acceleration

More Accurate Testing

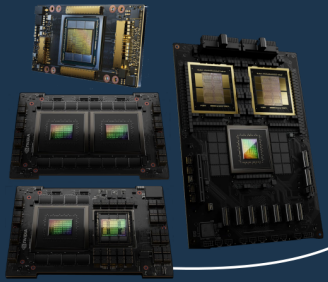
More bugs found with simulation

Huge Error Logs

Hard to Interpret

Design Copilot Assistant

Chip Design and Simulation



Chip Simulation

Simulation Acceleration

More Accurate Testing

More bugs found with simulation

Hard to Interpret

Huge Error Logs

Design Copilot Assistant

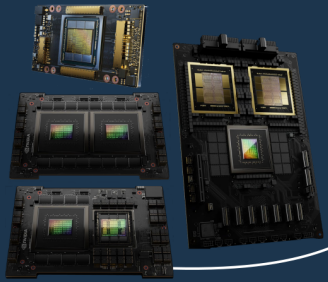
Does not find the solution

Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Design and Simulation



Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Simulation

Simulation Acceleration

More Accurate Testing

More bugs found with simulation

Hard to Interpret

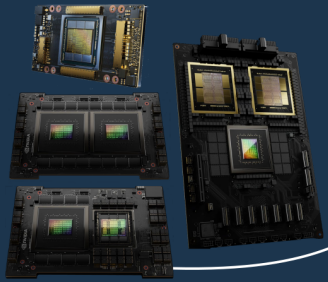
Huge Error Logs

Design Copilot Assistant

But helps engineers to find where the problem is

Does not find the solution

Chip Design and Simulation



Complex Chips

16 months to produce a chip

3 months to print it for real tests

Chip Simulation

Simulation Acceleration

More Accurate Testing

More bugs found with simulation

Hard to Interpret

Huge Error Logs

Design Copilot Assistant

But helps engineers to find where the problem is

From 2 days to 10 min to find most of problems location

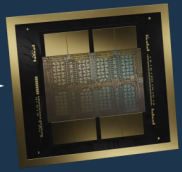
Does not find the solution

Conclusion

Conclusion

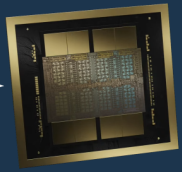
Titan 20 PFlops (2012)
7 MW

Blackwell 20 PFlops (2024)
1 kW



Conclusion

Titan 20 PFlops (2012)
7 MW



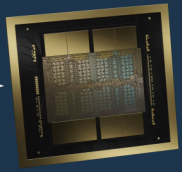
Blackwell 20 PFlops (2024)
1 kW

Need to be used efficently

Conclusion

Titan 20 PFlops (2012)
7 MW

Blackwell 20 PFlops (2024)
1 kW



Need to be used efficently

The **Work** has been **done** for **standards**

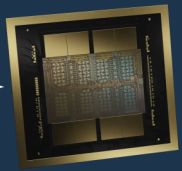
NumPy SciPy pandas scikit learn dmlc XGBoost JAX HDF Parquet

CPU + GPU scaling up to a full computing center

Conclusion

Titan 20 PFlops (2012)
7 MW

Blackwell 20 PFlops (2024)
1 kW



Need to be used efficently

The **Work** has been **done** for **standards**

CPU + GPU scaling up to a full computing center

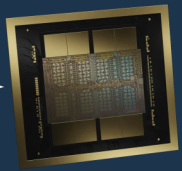
Hardware selling => efficient backends

Open Source Community

Conclusion

Titan 20 PFlops (2012)
7 MW

Blackwell 20 PFlops (2024)
1 kW



Need to be used efficently

We have to use standards

The **Work** has been **done** for **standards**

NumPy SciPy pandas scikit-learn dmlc XGBoost JAX HDF Parquet

CPU + GPU scaling up to a full computing center

NVIDIA intel

Hardware selling => efficient backends

Open Source Community

Link to a big summary

GTC 2024 notes from Reprises website