Laboratoire d'Annecy de Physique des Particules

Retour GTC 2022 : GPU Technical Conference

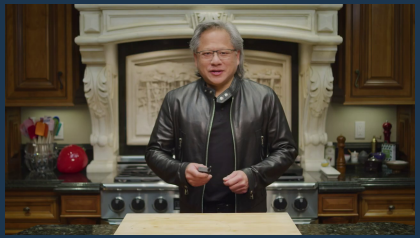
Pierre Aubert

ESCAPE
cherenkov telescope array
cnrs
UNIVERSITÉ SAVOIE MONT BLANC

Introduction

GTC 2022 `https://gtc22.event.nvidia.com/`

This is not possible to summarize **987** talks !



Keynotes video link

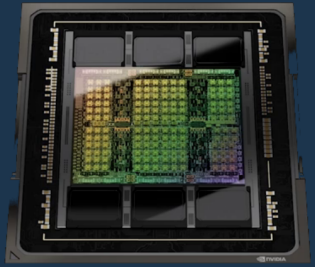
Only focus on :

- **Hardwares** evolution
- **Compilers** evolution
- **Programming Languages** evolution
- **Job Communication**
- **Data Compression/Decompression**
- **Machine Learning** / **Deep Learning** evolution and use
- How all (will) work together (with a bit of **Forum ORAP**)

Hardware evolution

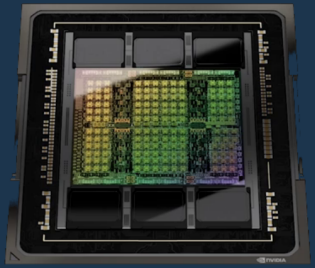
Hardware evolution

H100 (Hopper)



Hardware evolution

H100 (Hopper)

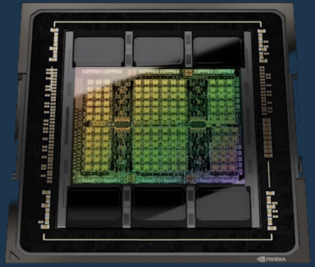


H100 SXM5 and PCie

Hardware evolution

4 nm (A100 7 nm)

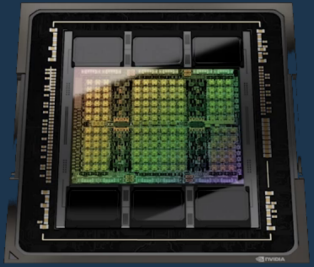
H100 (Hopper)



H100 SXM5 and PCie

Hardware evolution

H100 (Hopper)



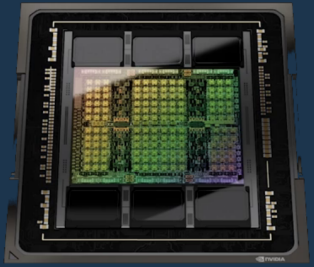
H100 SXM5 and PCie

4 nm (A100 7 nm)

Compute Capabilites 9.0

Hardware evolution

H100 (Hopper)



H100 SXM5 and PCie

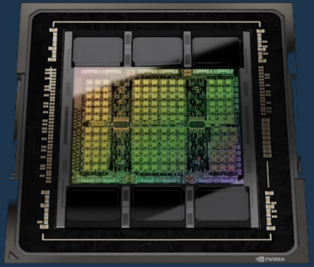
4 nm (A100 7 nm)

Compute Capabilites 9.0

132 SMs (tensor core Gen4) 8448 Cores

Hardware evolution

H100 (Hopper)



H100 SXM5 and PCie

4 nm (A100 7 nm)

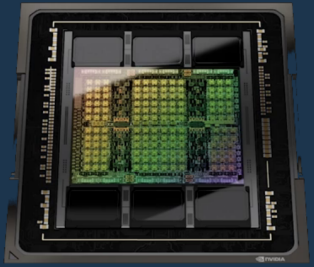
Compute Capabilites 9.0

132 SMs (tensor core Gen4) 8448 Cores

80 GB DRAM

Hardware evolution

H100 (Hopper)



H100 SXM5 and PCie

4 nm (A100 7 nm)

Compute Capabilites 9.0

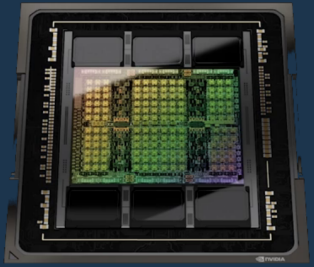
132 SMs (tensor core Gen4) 8448 Cores

80 GB DRAM

3 TB/s Bandwidth (HBM3)

Hardware evolution

H100 (Hopper)



H100 SXM5 and PCie

4 nm (A100 7 nm)

Compute Capabilites 9.0

132 SMs (tensor core Gen4) 8448 Cores

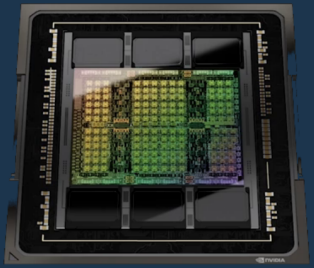
80 GB DRAM

3 TB/s Bandwidth (HBM3)

NVLink Gen 4 (900 GB/s)

Hardware evolution

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

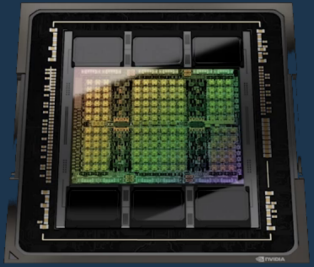
3 TB/s Bandwidth (HBM3)

NVLink Gen 4 (900 GB/s)

MIG Gen 2 with confidential computing
(all instances have Image/Video decoding)

Hardware evolution

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

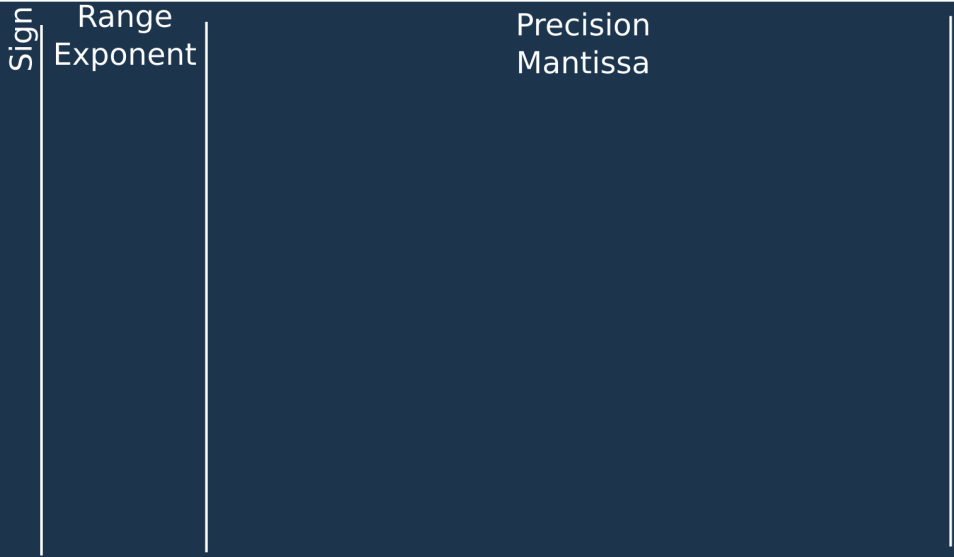
3 TB/s Bandwidth (HBM3)

NVLink Gen 4 (900 GB/s)

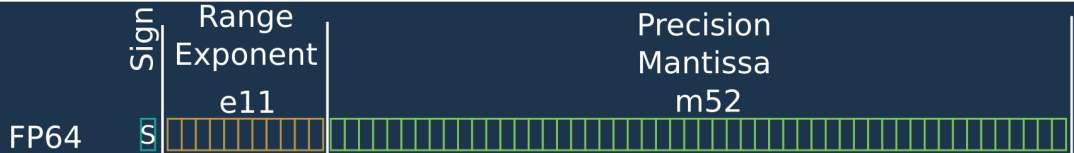
MIG Gen 2 with confidential computing
(all instances have Image/Video decoding)

A lot a computing precisions

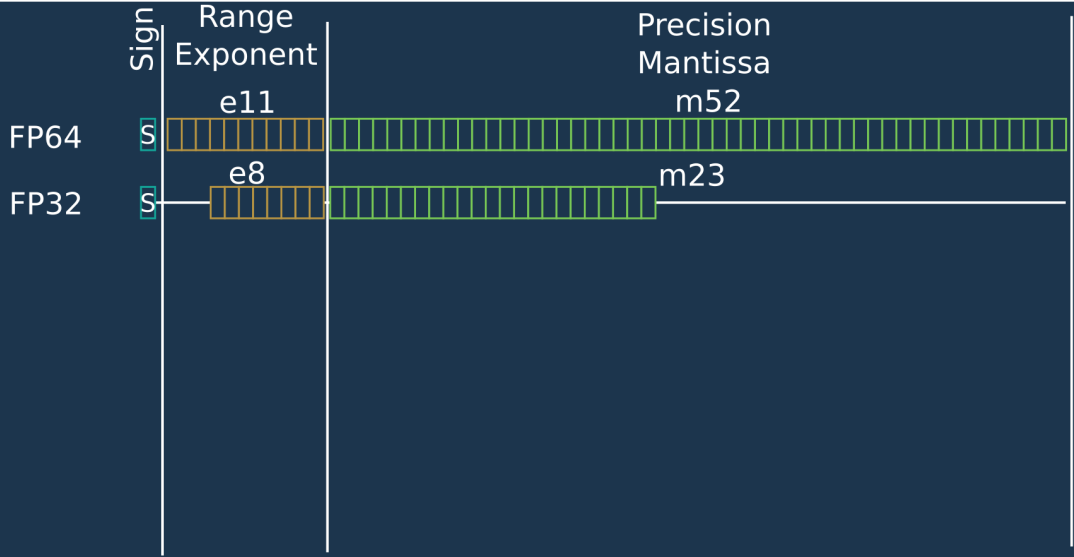
Computing Precision



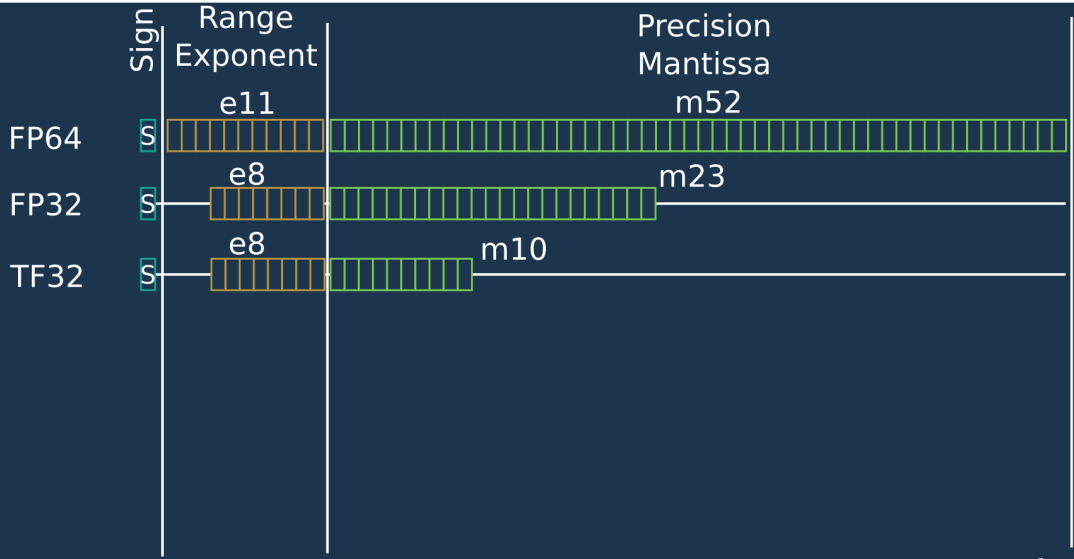
Computing Precision



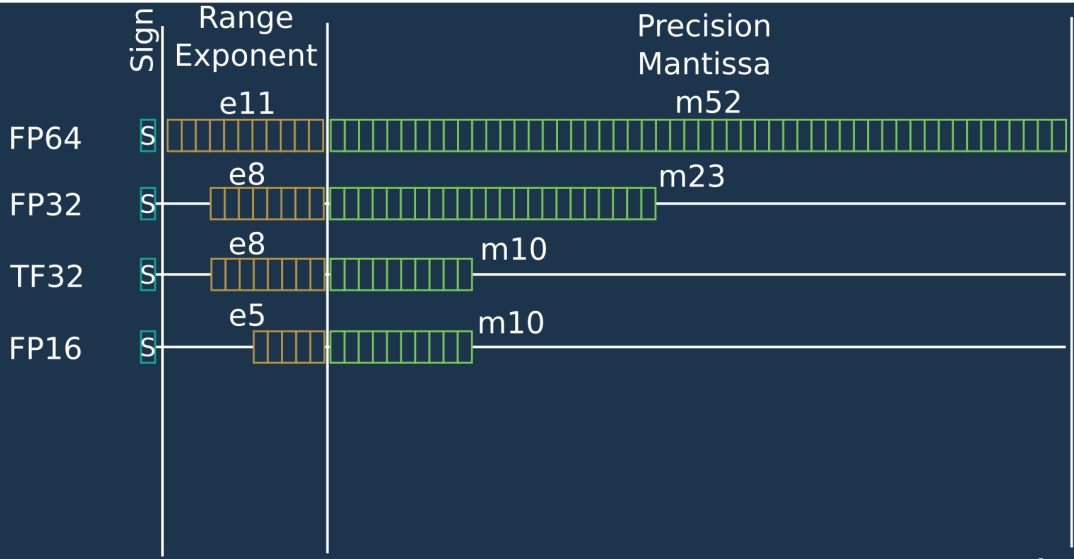
Computing Precision



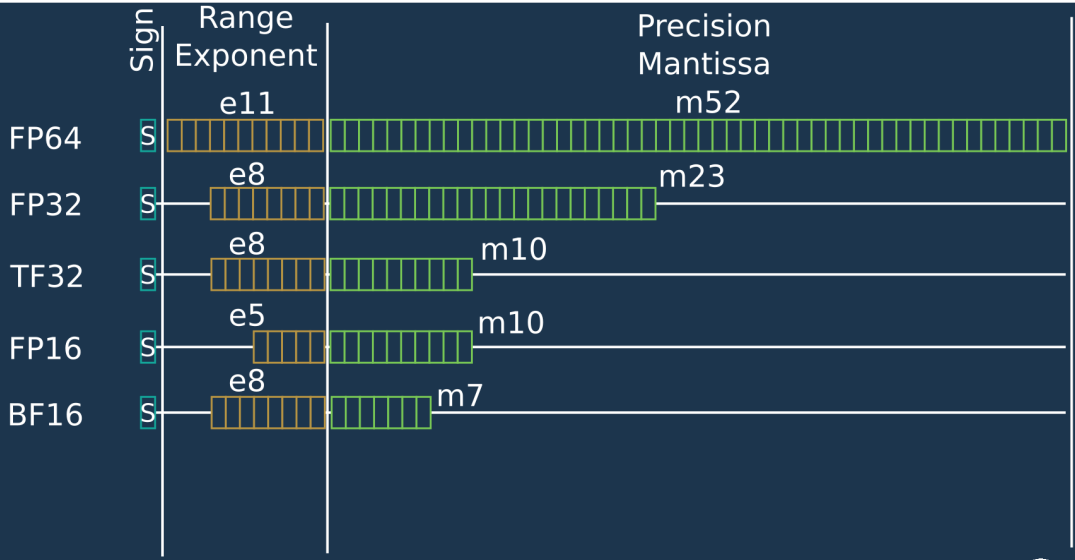
Computing Precision



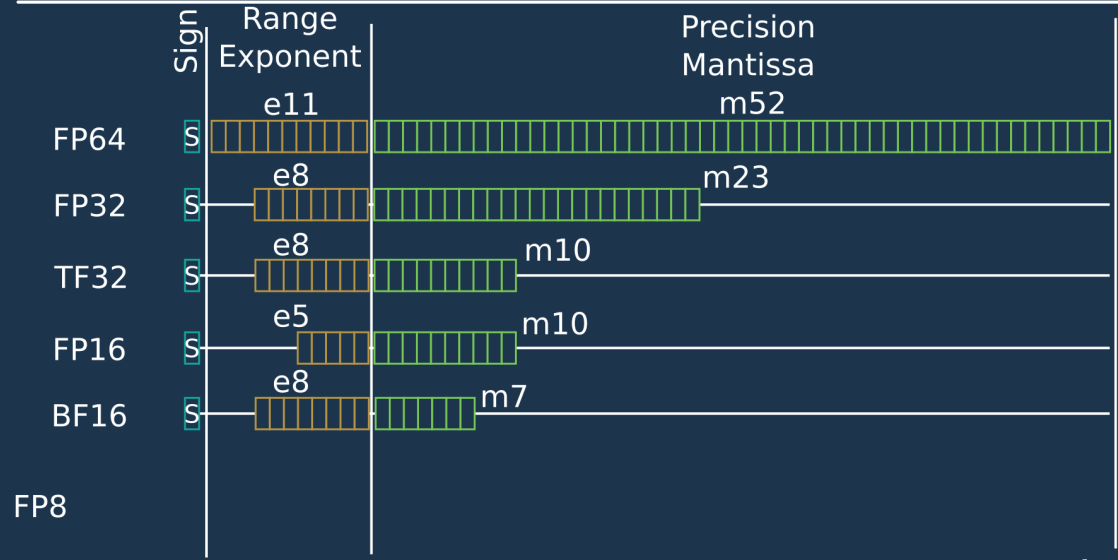
Computing Precision



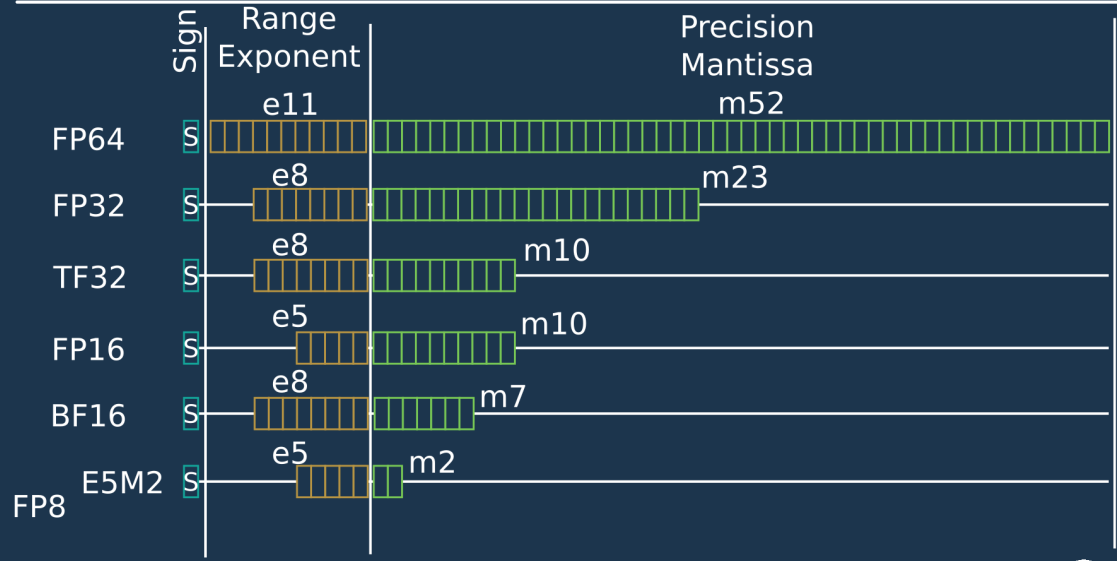
Computing Precision



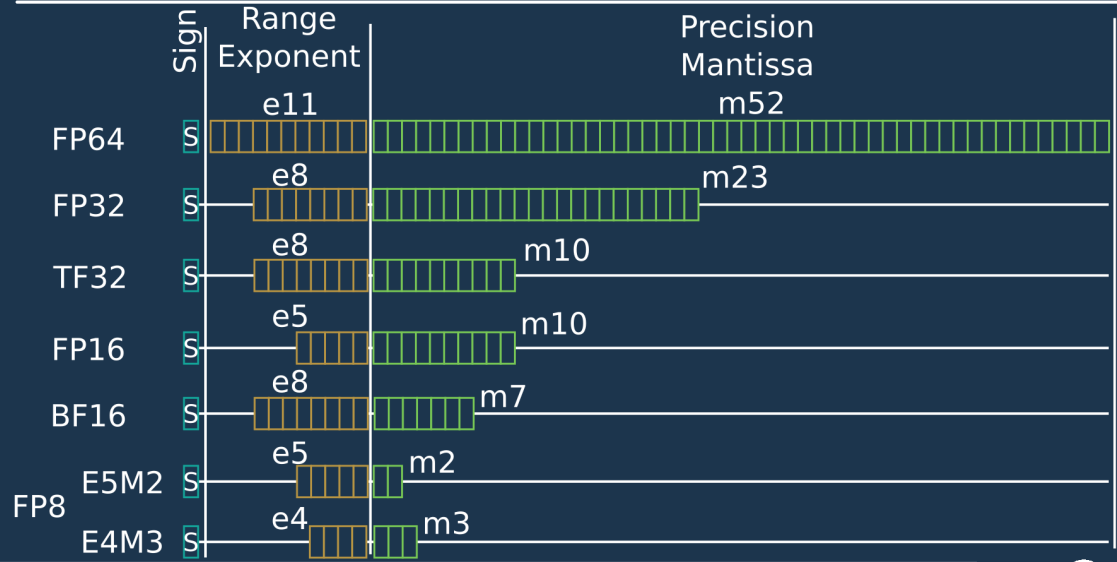
Computing Precision



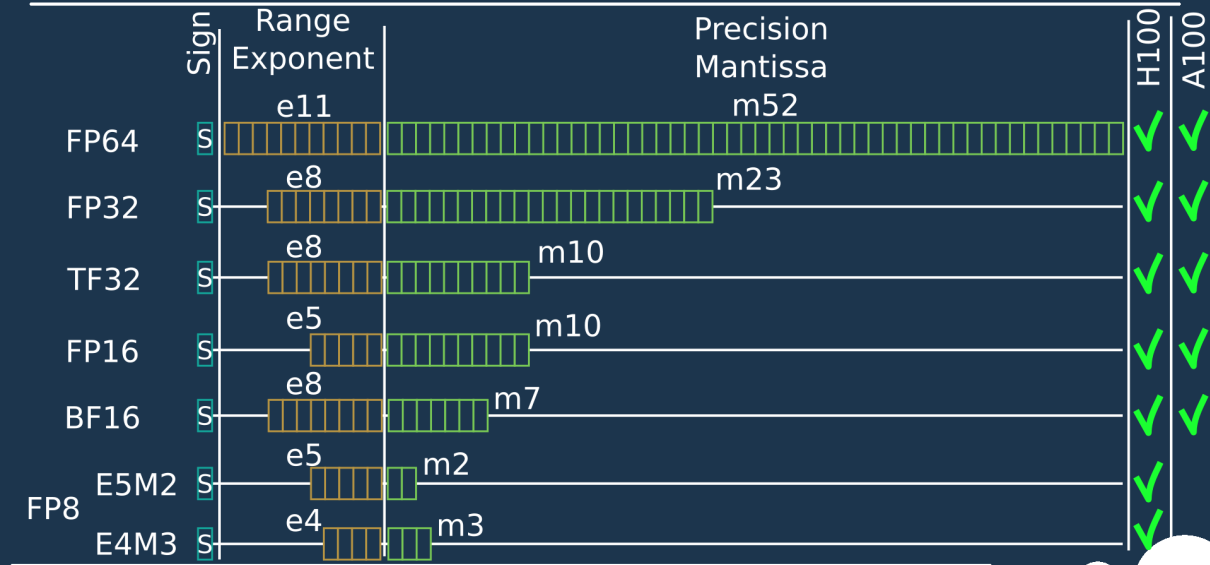
Computing Precision



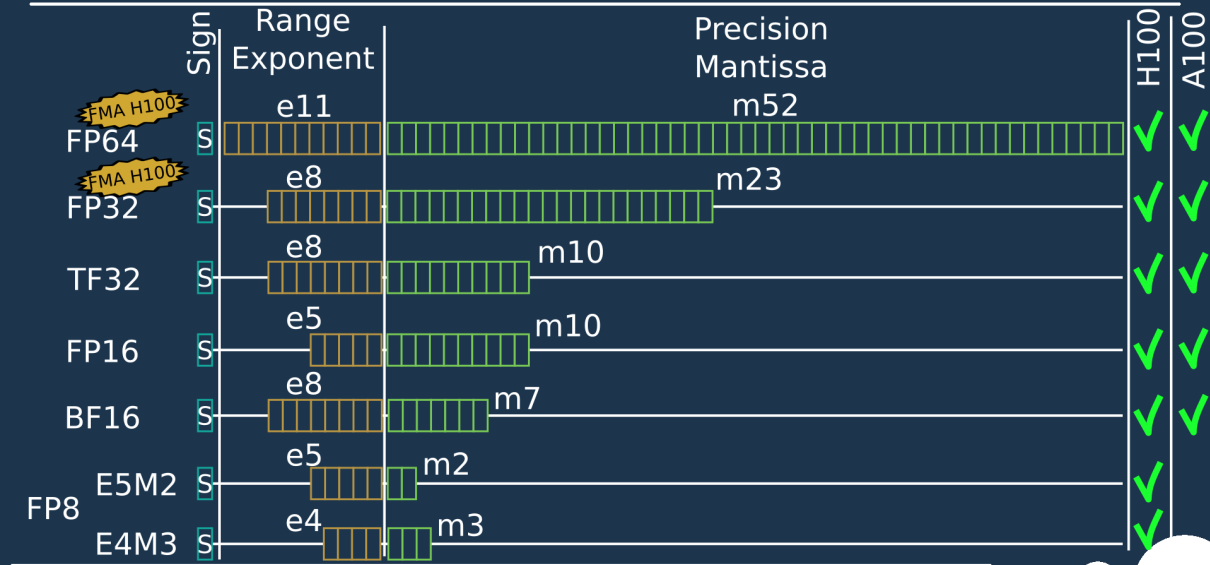
Computing Precision



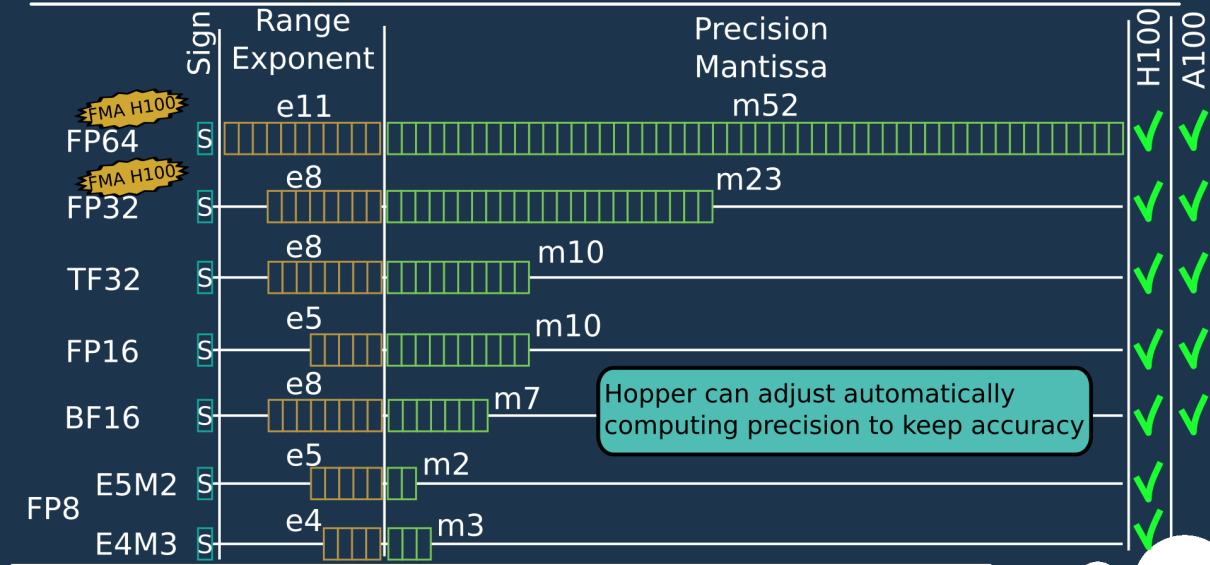
Computing Precision



Computing Precision



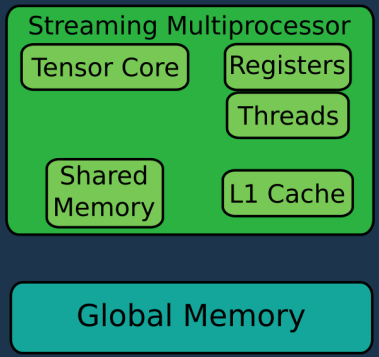
Computing Precision



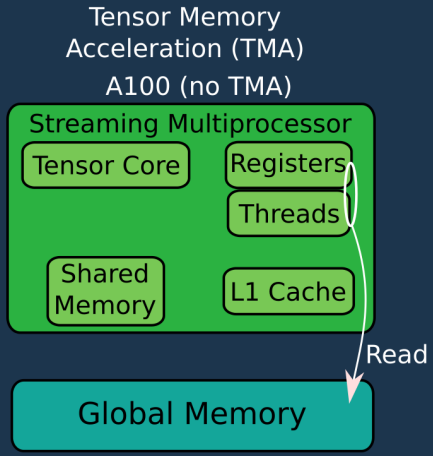
Thread acceleration

Thread acceleration

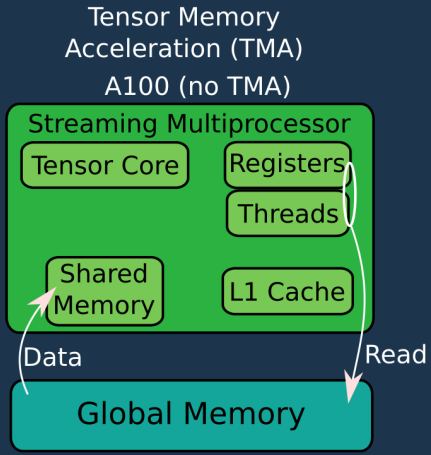
Tensor Memory Acceleration (TMA)



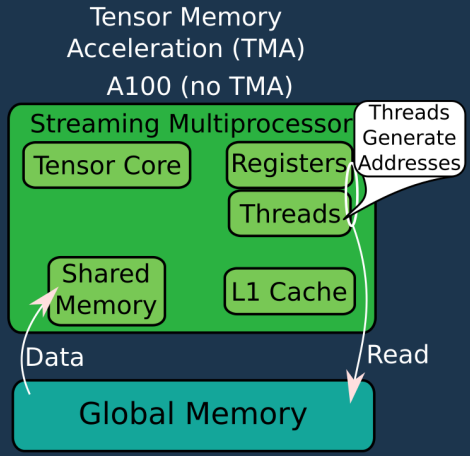
Thread acceleration



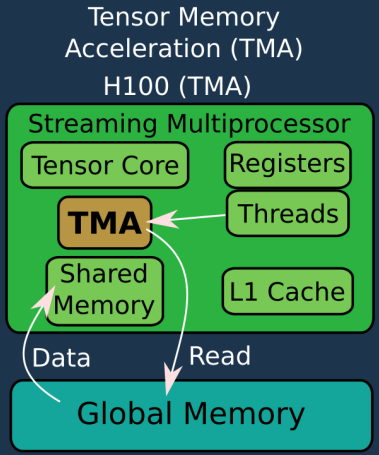
Thread acceleration



Thread acceleration

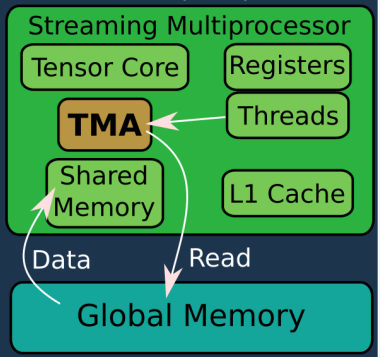


Thread acceleration



Thread acceleration

Tensor Memory Acceleration (TMA)

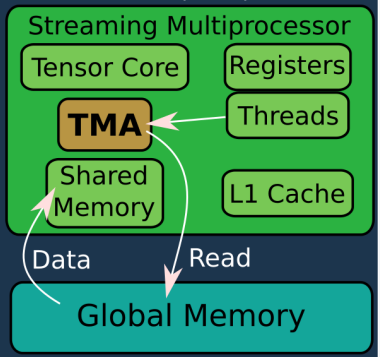


Distributed shared memory

Thread acceleration

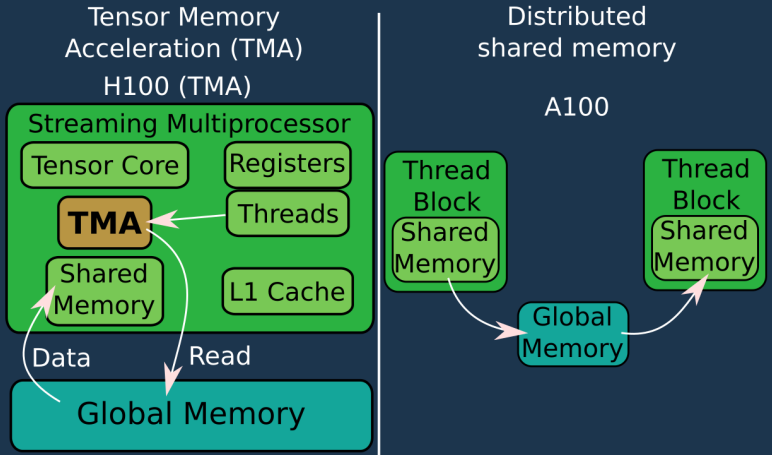
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

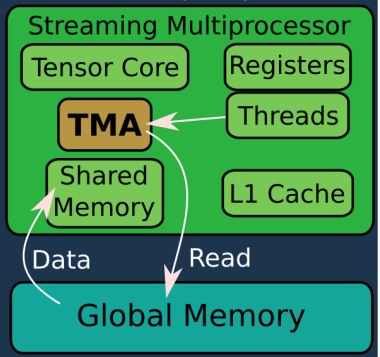
Thread acceleration



Thread acceleration

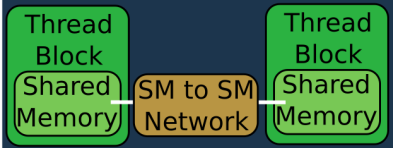
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

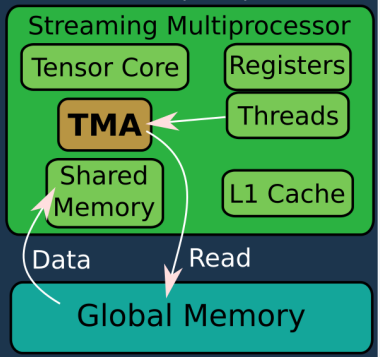
H100



Thread acceleration

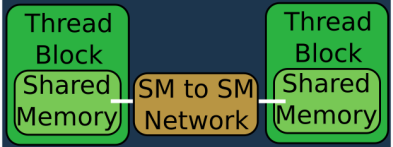
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

H100

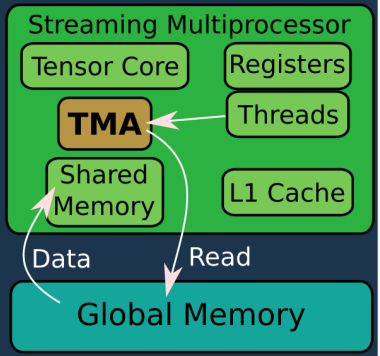


Asynchronous Transaction Barrier

Thread acceleration

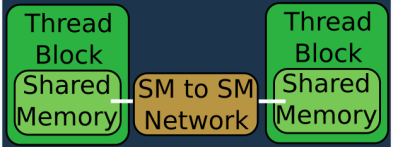
Tensor Memory Acceleration (TMA)

H100 (TMA)



Distributed shared memory

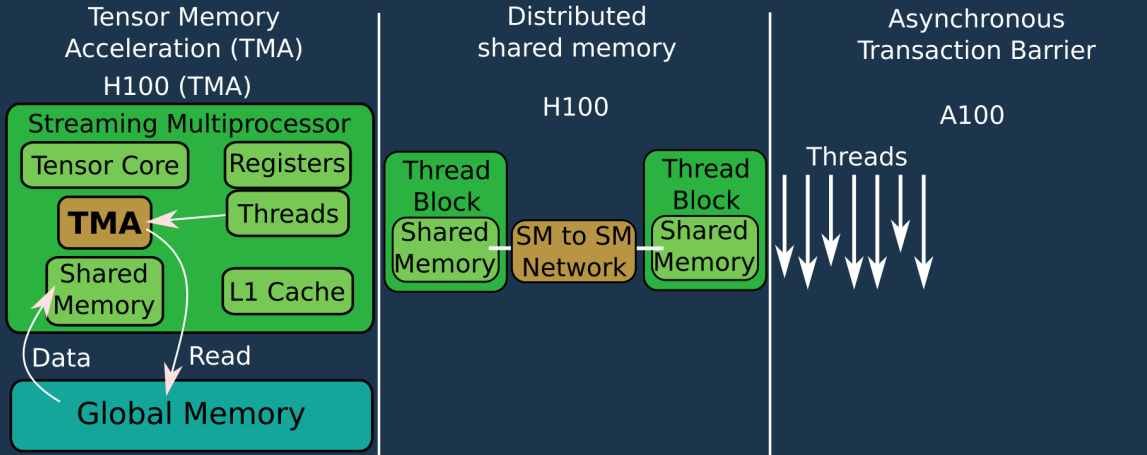
H100



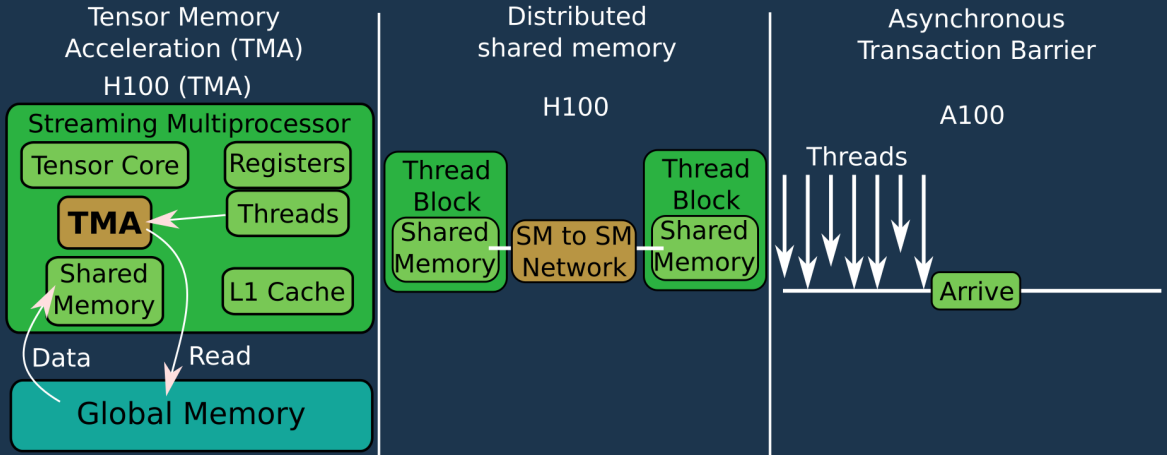
Asynchronous Transaction Barrier

A100

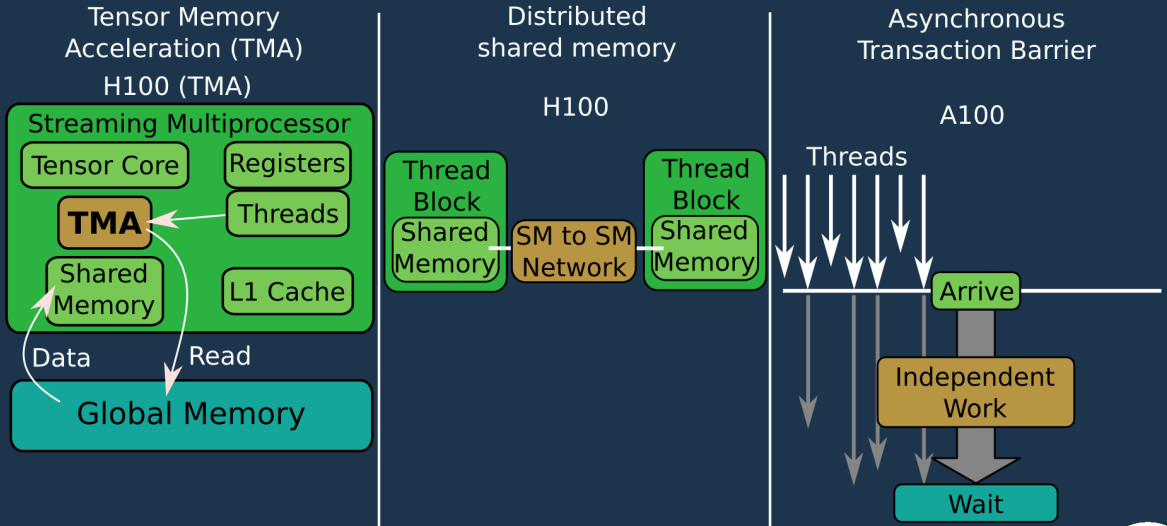
Thread acceleration



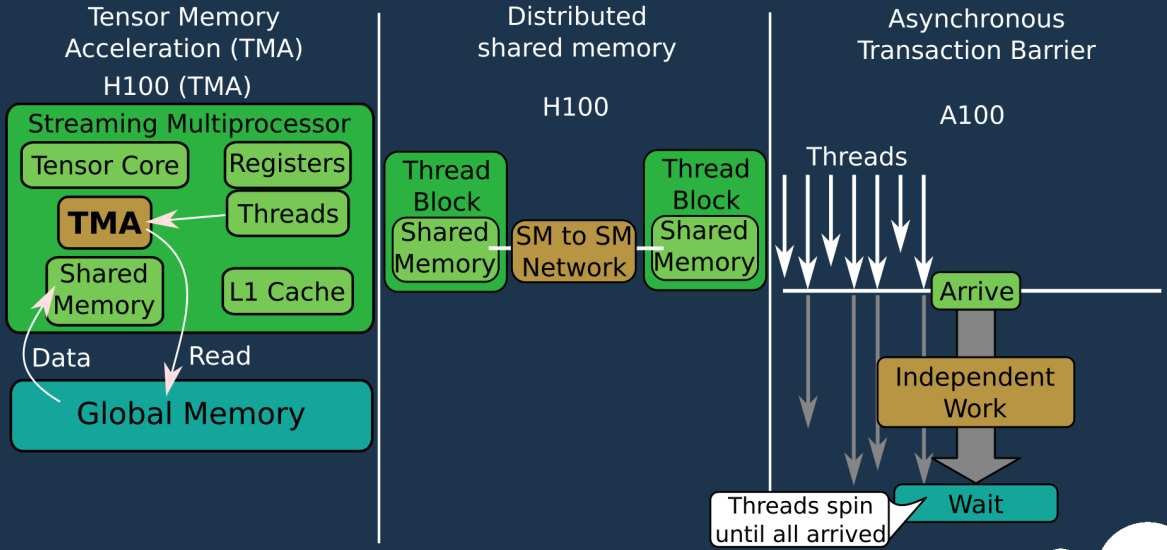
Thread acceleration



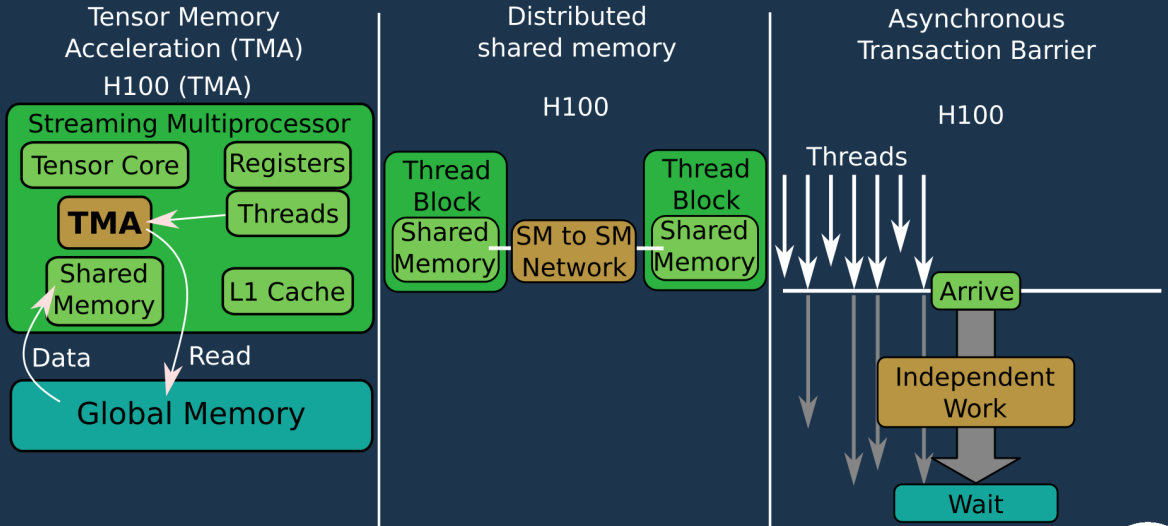
Thread acceleration



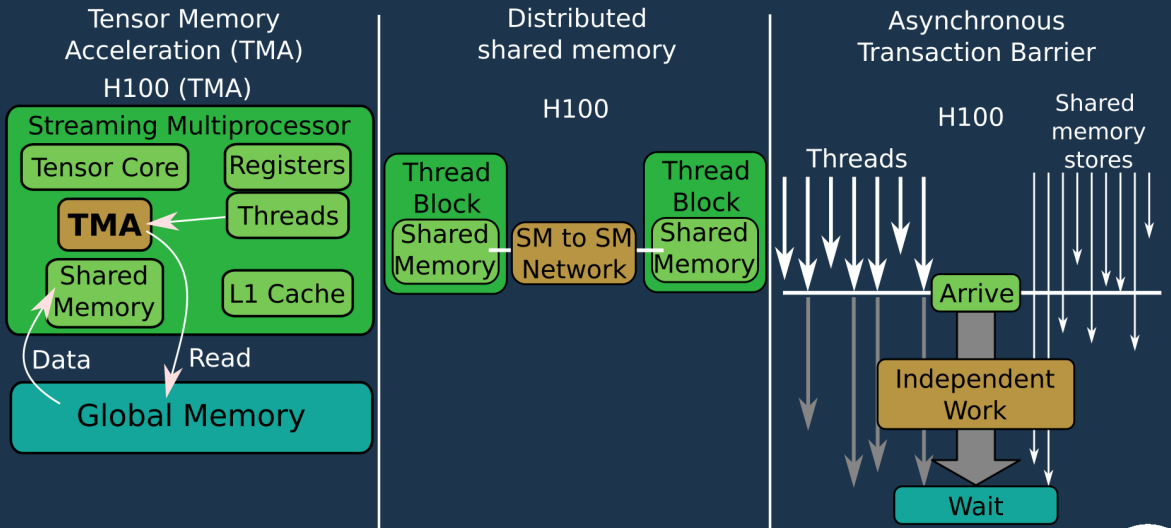
Thread acceleration



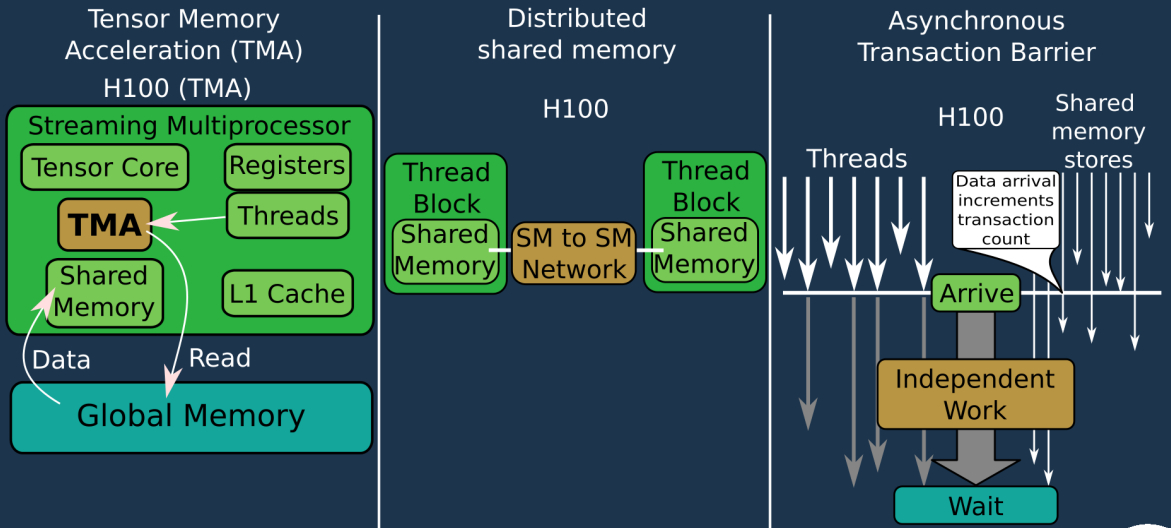
Thread acceleration



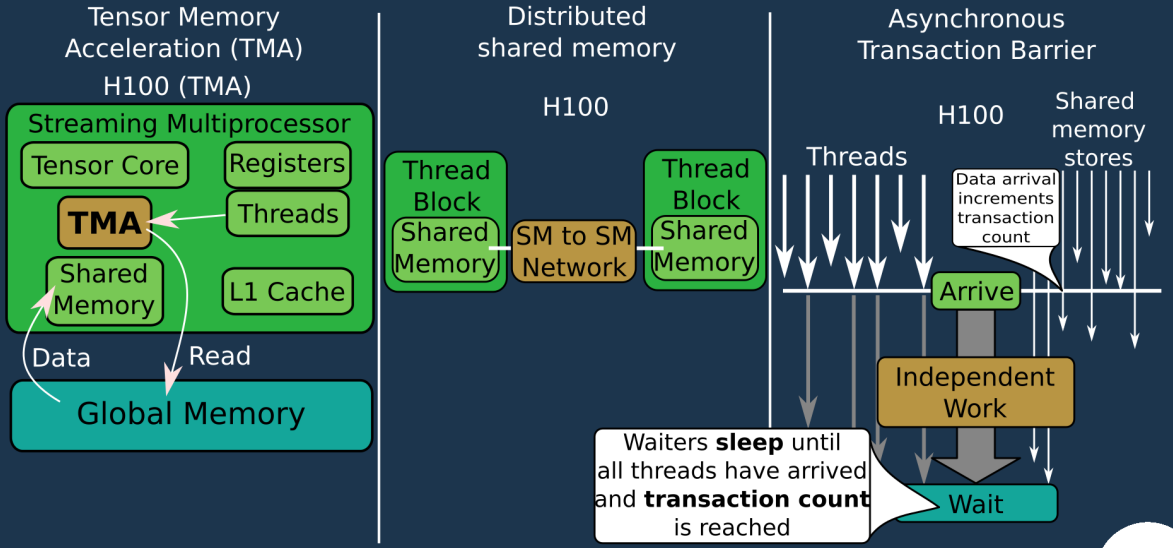
Thread acceleration



Thread acceleration



Thread acceleration



New intrinsic functions

New intrinsic functions

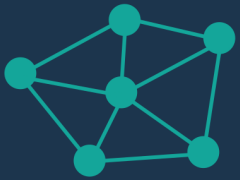
Dynamic Programming Acceleration (DPX) : recursive algorithms

New intrinsic functions

Dynamic Programming Acceleration (DPX) : recursive algorithms

Floyd-Warshall Algorithm

Route optimization

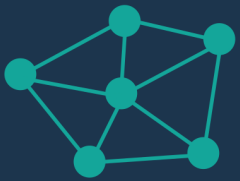


New intrinsic functions

Dynamic Programming Acceleration (DPX) : recursive algorithms

Floyd-Warshall Algorithm

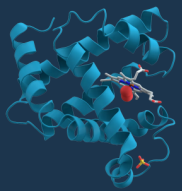
Route optimization



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

Pattern Matching

- DNA sequence alignment
- Protein classification
- Protein folding

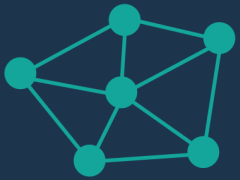


New intrinsic functions

Dynamic Programming Acceleration (DPX) : recursive algorithms

Floyd-Warshall Algorithm

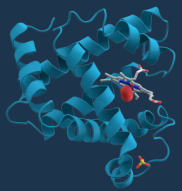
Route optimization



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

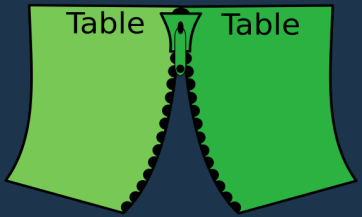
Pattern Matching

- DNA sequence alignment
- Protein classification
- Protein folding



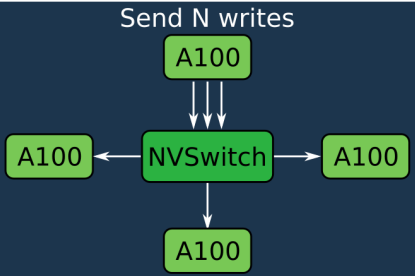
SQL Query optimisation

Join in optimal order

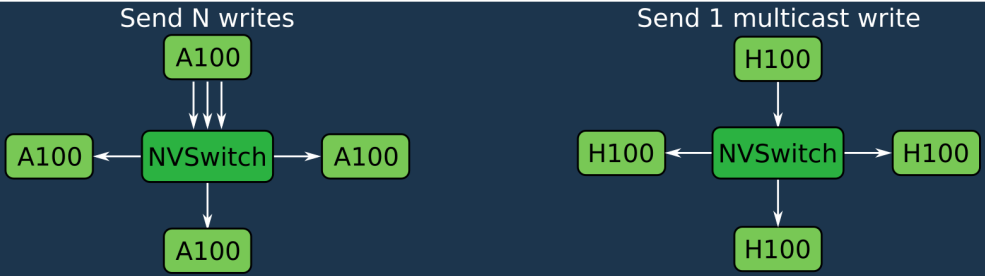


NVLink/NVSwitch acceleration

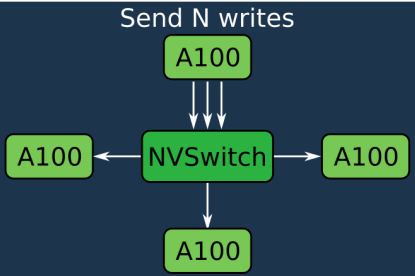
NVLink/NVSwitch acceleration

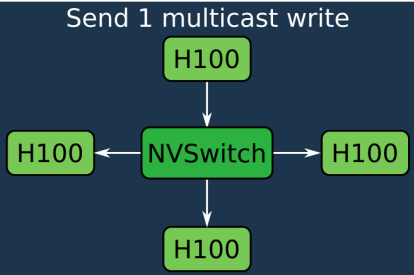


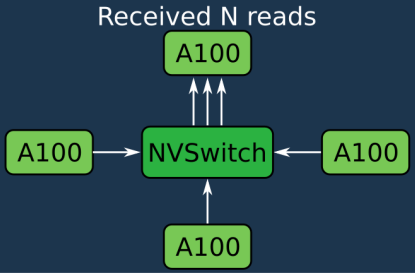
NVLink/NVSwitch acceleration



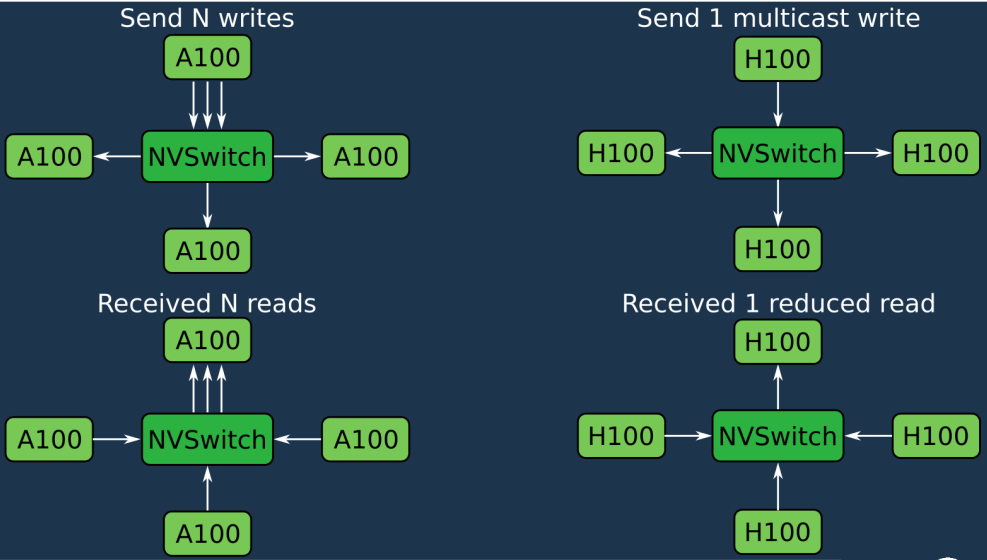
NVLink/NVSwitch acceleration







NVLink/NVSwitch acceleration



Thread block cluster

Thread block cluster

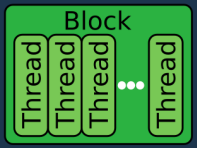
A100 (108 SMs)

Thread block cluster

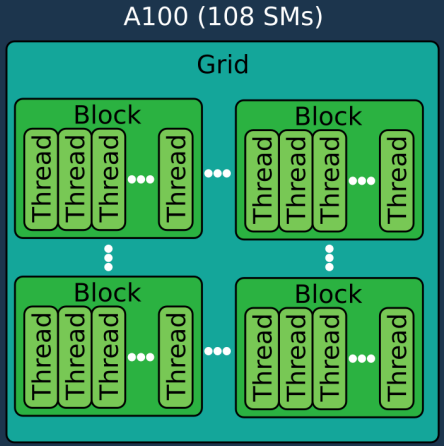
A100 (108 SMs)

Thread block cluster

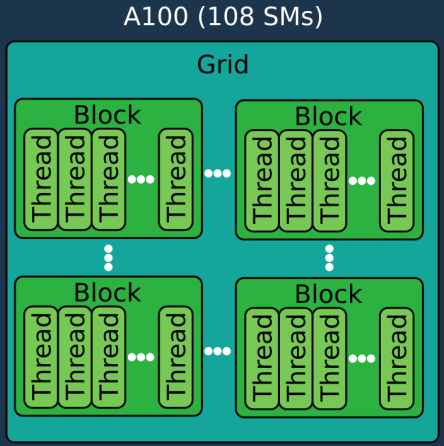
A100 (108 SMs)



Thread block cluster



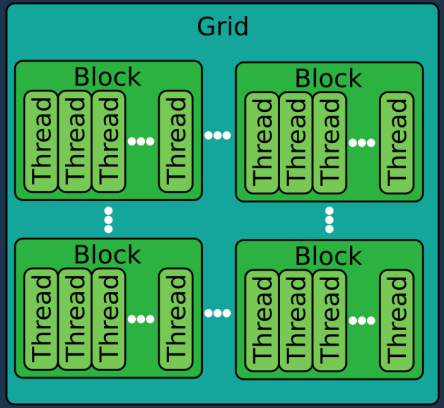
Thread block cluster



1 Block ~ 1 SM < 1% A100

Thread block cluster

A100 (108 SMs)

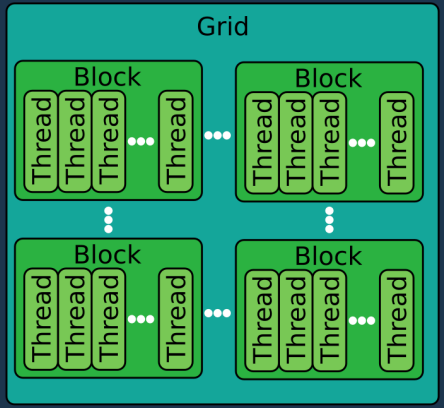


1 Block ~ 1 SM < 1% A100

H100 (132 SMs)

Thread block cluster

A100 (108 SMs)

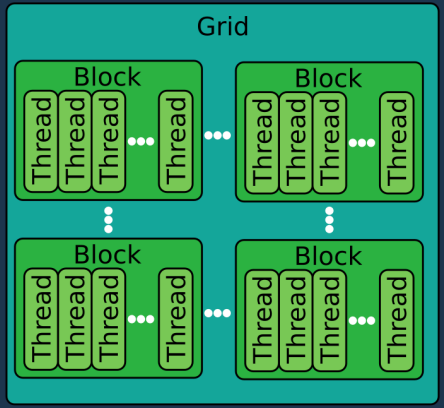


1 Block ~ 1 SM < 1% A100

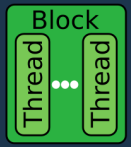
H100 (132 SMs)

Thread block cluster

A100 (108 SMs)



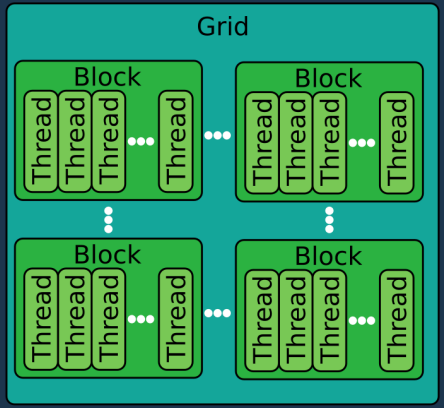
H100 (132 SMs)



1 Block ~ 1 SM < 1% A100

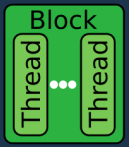
Thread block cluster

A100 (108 SMs)



1 Block ~ 1 SM < 1% A100

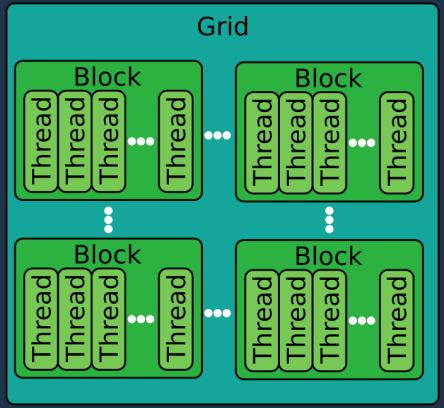
H100 (132 SMs)



1 Block ~ 1 SM < 0.8 % H100

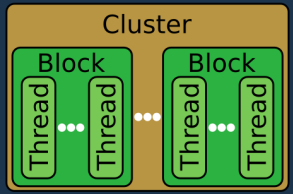
Thread block cluster

A100 (108 SMs)



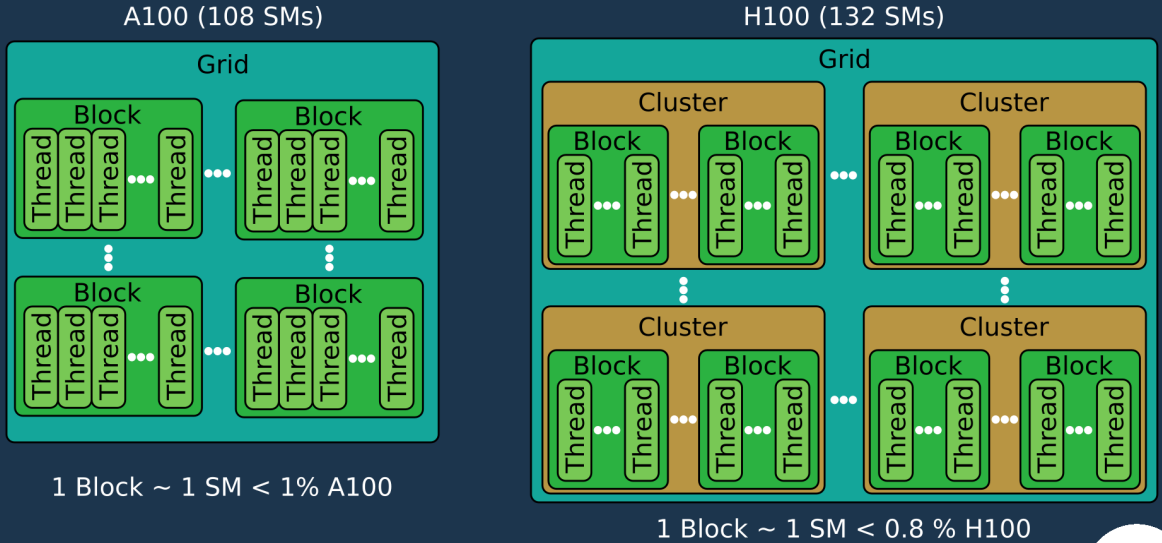
1 Block ~ 1 SM < 1% A100

H100 (132 SMs)



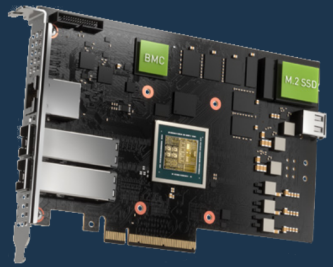
1 Block ~ 1 SM < 0.8 % H100

Thread block cluster



Bluefield 3

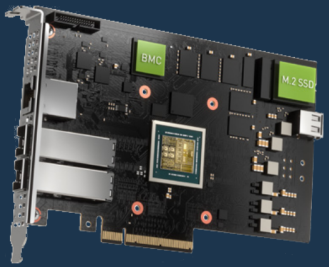
DPU : Data Processing Unit



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

Bluefield 3

DPU : Data Processing Unit

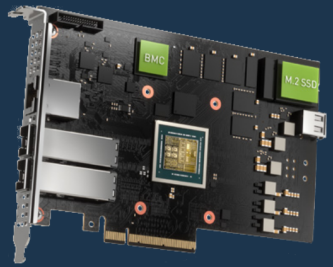


Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

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

Bluefield 3

DPU : Data Processing Unit



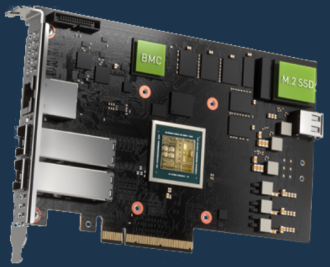
Bluefield 3 DPU – 2x 200Gb/s
FHHL form factor

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

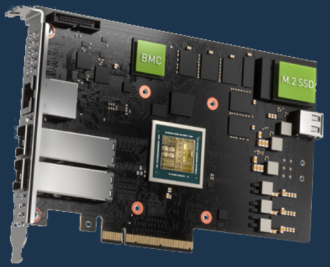
1, 2, 4 ports with up to 400Gb/s connectivity
(Ethernet or NDR InfiniBand)

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)

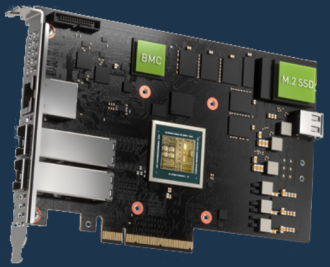
16 GB on-board DDR5 memory

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

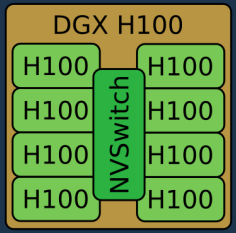
1 GbE out-of-band management port

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

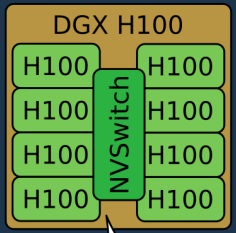
Form factors: HHHL, FHHL

DGX H100 / DGX POD H100

DGX H100 / DGX POD H100

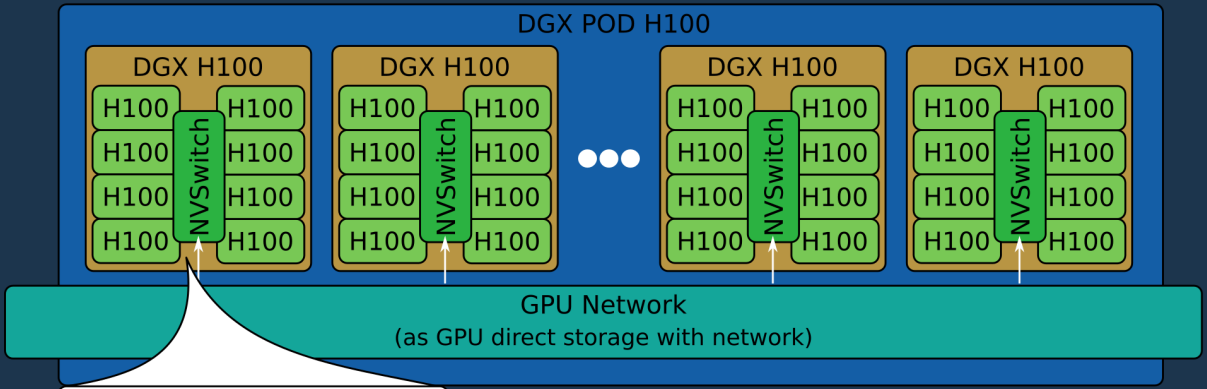


DGX H100 / DGX POD H100



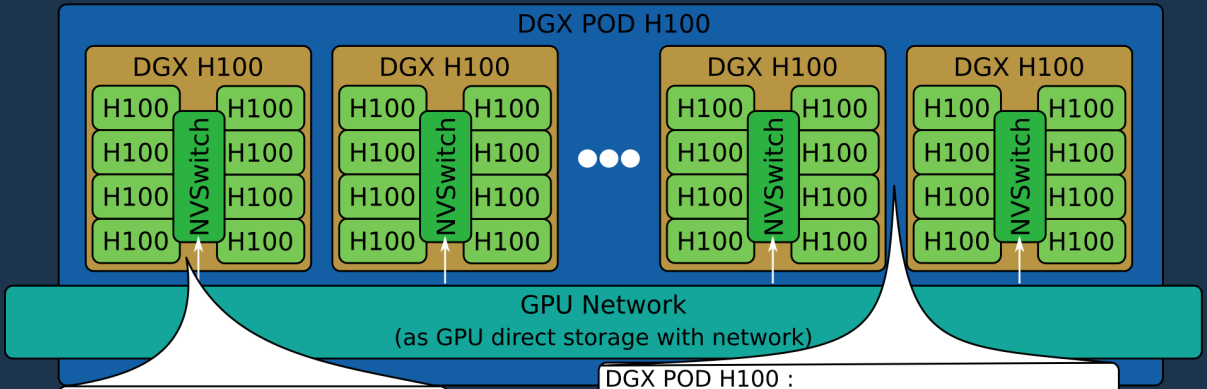
DGX H100 :
- 8 H100
- 32 PFlop (AI performance)
- 640 GB HBM3 Memory
- 24 TB/s memory Bandwidth

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

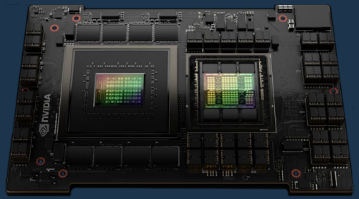
DGX POD H100 :
- 32 DGX H100
- 1 EFlop (AI performance)
- 20 TB HBM3 Memory
- 70 TB/s memory Bandwidth
- 192 TFlops of Sharp in-Network compute

Grace architecture 2023

Grace Supership

Grace architecture 2023

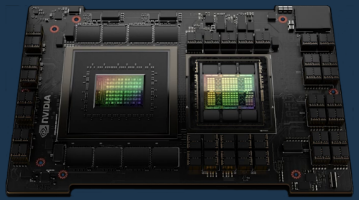
Grace Supership



Grace Hopper

Grace architecture 2023

Grace Supership

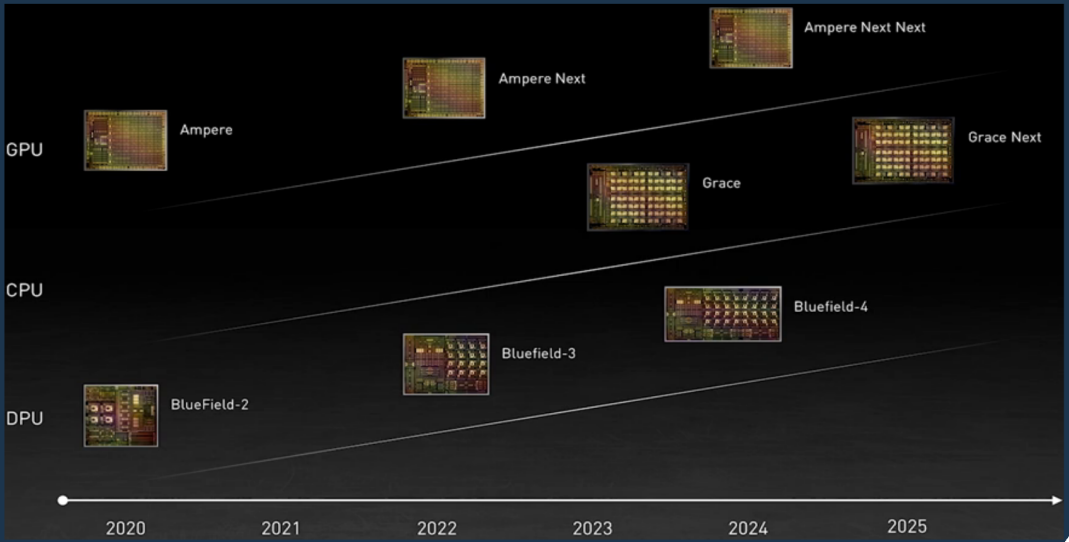


Grace Hopper

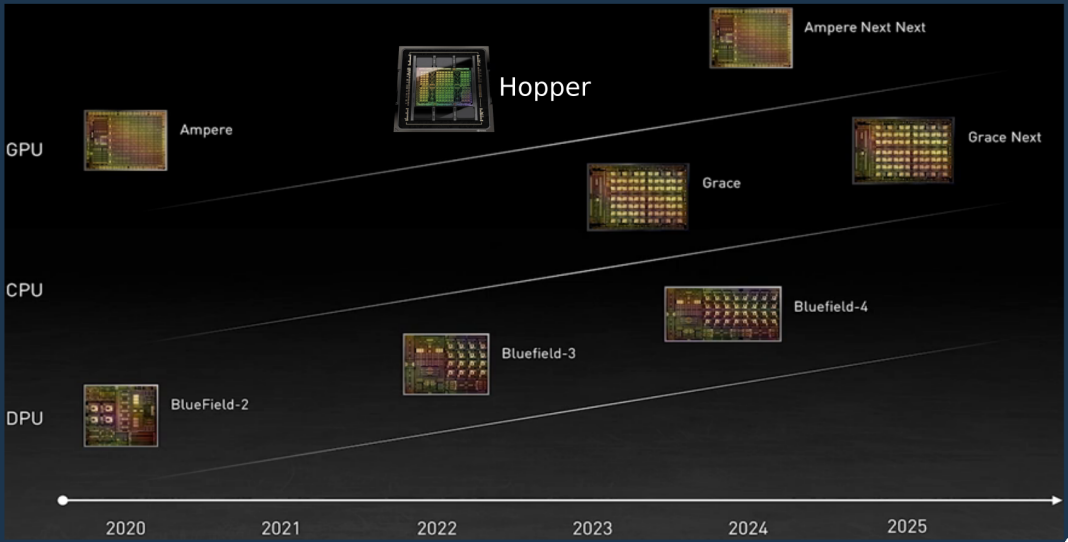
1906 - 1992

Grace architecture 2023

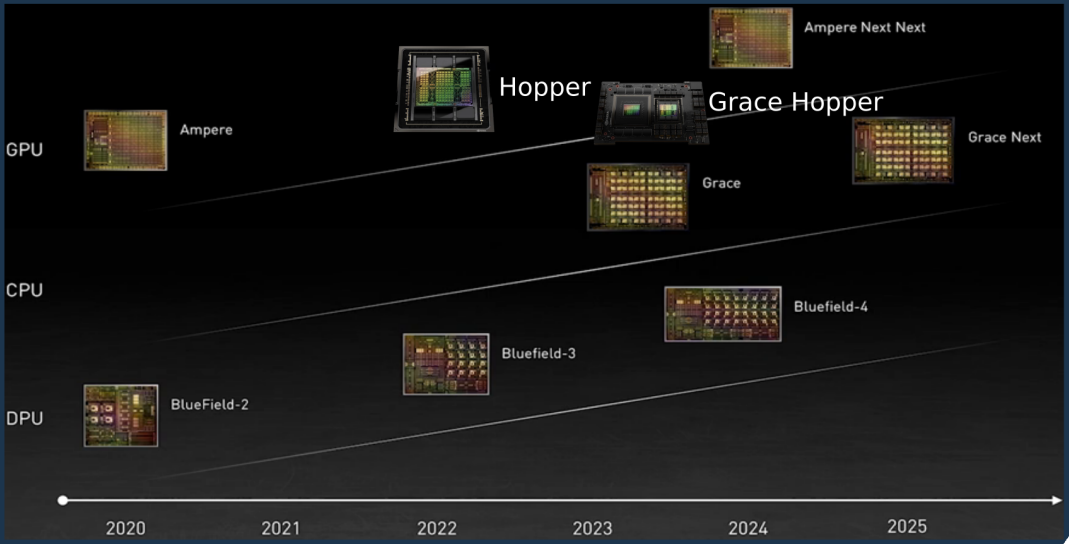
NVidia roadmap



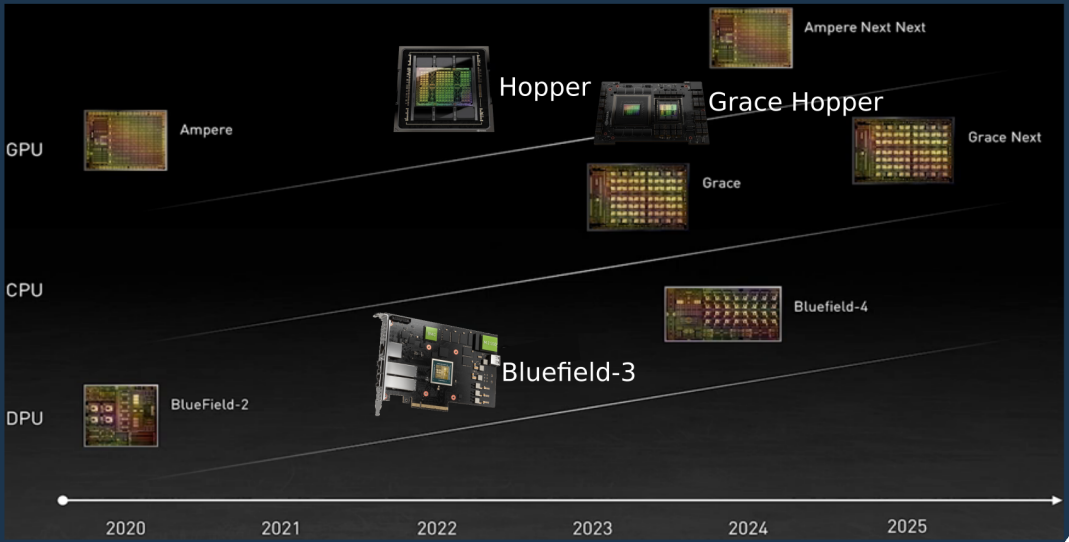
NVidia roadmap



NVidia roadmap



NVidia roadmap



The Fermat Architecture

Fermat "Storage Attached Computing"

- Storage : **Flash**
- Computing : **ARM + FPGA**

The Fermat Architecture

Fermat "Storage Attached Computing"

- Storage : **Flash**
- Computing : **ARM + FPGA**

Extract & Label atmospheric and river feature from **8 TB** Timed Raw **NOAA Satellite**

Dell - Xeon Server :
- **2** Intel Xeon Gold 6240 @ 2.6 GHz
- **18** cores, **36** threads, 24.75 MB cache each
- **384 GB** DDR4 @ 2933 MHz
- Storage **RAID 5**, **16 x 7.67 TB** SSD
- OS : CentOS Linux release 7.6.1810

Fermat :
- **9** FPGAs
- **192 GB** DDR4 DRAM
- **8 TB** NAND Flash

Time to complete : **44 minutes**

Time to complete : **22 seconds (118x)**

Compilers news

NVC++

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 : 
- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 : 
- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X : 
- Executors
- Linear algebra

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 : 
- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X : 
- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 : 
- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X : 
- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v)
(ex: a+b)

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v)
(ex: a+b)

More Parallel compilation coming

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v)
(ex: a+b)

More Parallel compilation coming

Ongoing migration **LLVM 7**

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v)
(ex: a+b)

More Parallel compilation coming

Ongoing migration **LLVM 7**

C++20 :

- Module but no cuda call

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v)
(ex: a+b)

More Parallel compilation coming


Ongoing migration **LLVM 7**


C++20 :

- Module but no cuda call

NVFortran

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v) (ex: a+b)

More Parallel compilation coming


Ongoing migration **LLVM 7**


C++20 :

- Module but no cuda call

NVFortran

Uses **nvTENSOR**

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v) (ex: a+b)

More Parallel compilation coming


Ongoing migration **LLVM 7**


C++20 :

- Module but no cuda call

NVFortran

Uses **nvTENSOR**

Tensor Core available

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v) (ex: a+b)

More Parallel compilation coming


Ongoing migration **LLVM 7**


C++20 :

- Module but no cuda call

NVFortran

Uses **nvTENSOR**

Tensor Core available

Do Concurent :
- No function call
- Reduction not available yet

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v) (ex: a+b)

More Parallel compilation coming


Ongoing migration **LLVM 7**


C++20 :

- Module but no cuda call

NVFortran

Uses **nvTENSOR**

Tensor Core available

Do Concurent :
- No function call
- Reduction not available yet

Same Perf as **OpenACC**

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 : 
- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X : 
- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v) (ex: a+b)

More Parallel compilation coming 

Ongoing migration **LLVM 7** 

C++20 : 
- Module but no cuda call

NVFortran

Uses **nvTENSOR**

Tensor Core available

Do Concurent :
- No function call
- Reduction not available yet

Same Perf as **OpenACC**

Cuda Fortran not portable

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v) (ex: a+b)

More Parallel compilation coming


Ongoing migration **LLVM 7**


C++20 :

- Module but no cuda call

NVFortran

Uses **nvTENSOR**

Tensor Core available

Do Concurent :
- No function call
- Reduction not available yet

Same Perf as **OpenACC**

Cuda Fortran not portable

For now **NVidia** only :

- **Intel**, **Cray** ongoing

Compilers news

NVC++

C++17 :
- Combine **std::transform** and **std::reduce** in the same call

C++20 :
- Parallelism Thread concurency

C++23 :

- Range based parallel algorithm (mdspan, mdarray)
- Multi-Dim Array abstractions

C++2X :

- Executors
- Linear algebra

NVCC

Autodetection **GPU**

128 bits integers supported since version 11.6

Link Time Optimization (LTO)

Parallel compilation for different compute capabilities (-d -thread)

Implementation detail (-d -v) (ex: a+b)

More Parallel compilation coming

Ongoing migration **LLVM 7**

C++20 :

- Module but no cuda call

NVFortran

Uses **nvTENSOR**

Tensor Core available

Do Concurent :
- No function call
- Reduction not available yet

Same Perf as **OpenACC**

Cuda Fortran not portable

For now **NVidia** only :

- **Intel**, **Cray** ongoing

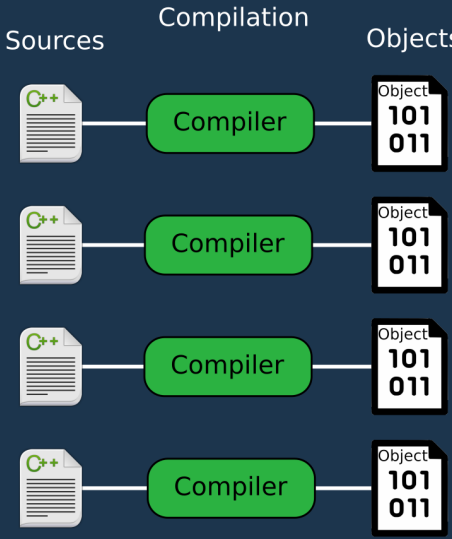
Fortran Forever :
- Modern Fortran >= **F18**

Link Time Optimization

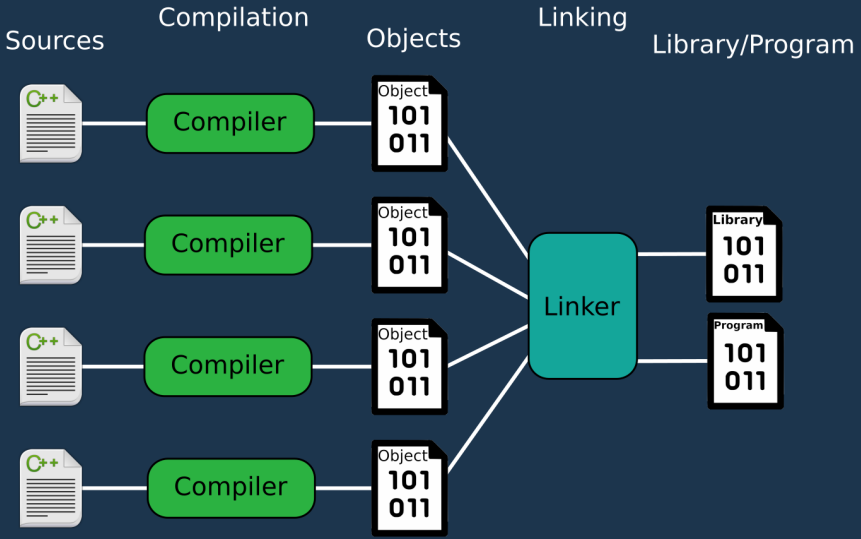
Link Time Optimization

Sources

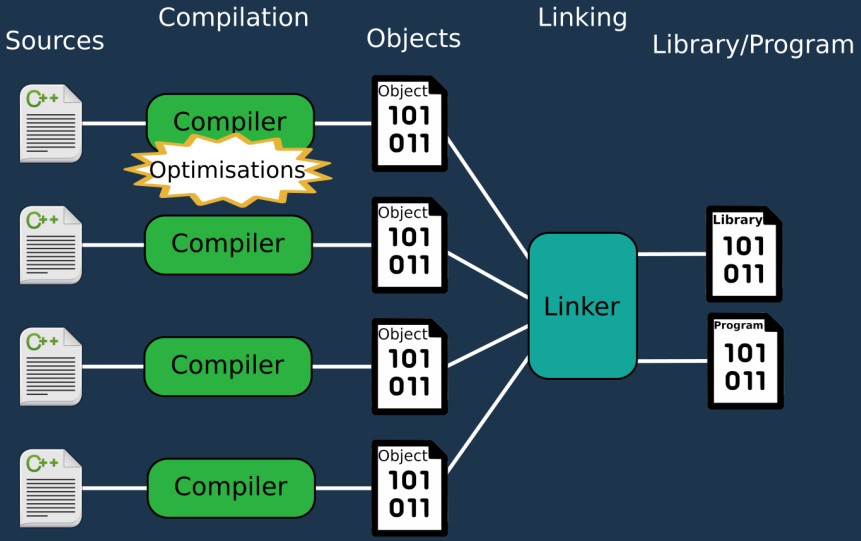
Link Time Optimization



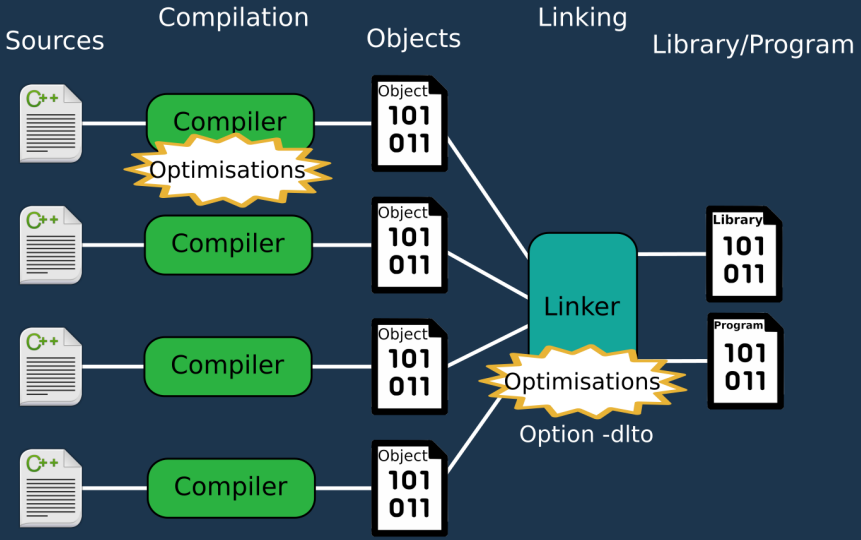
Link Time Optimization



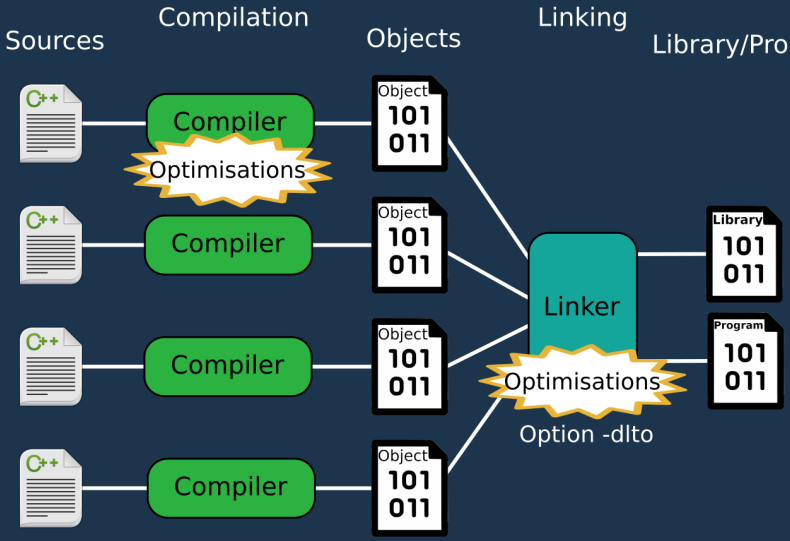
Link Time Optimization



Link Time Optimization

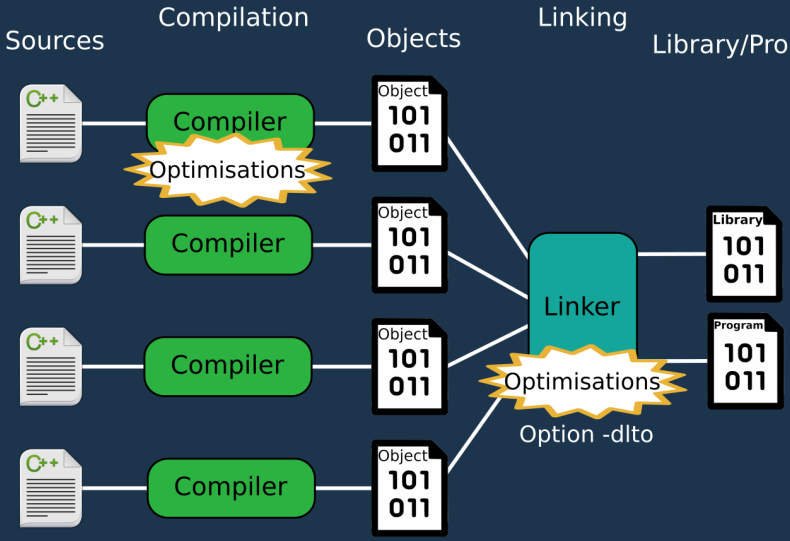


Link Time Optimization



LTO Optimisations :
- Remove dead code (even kernel)

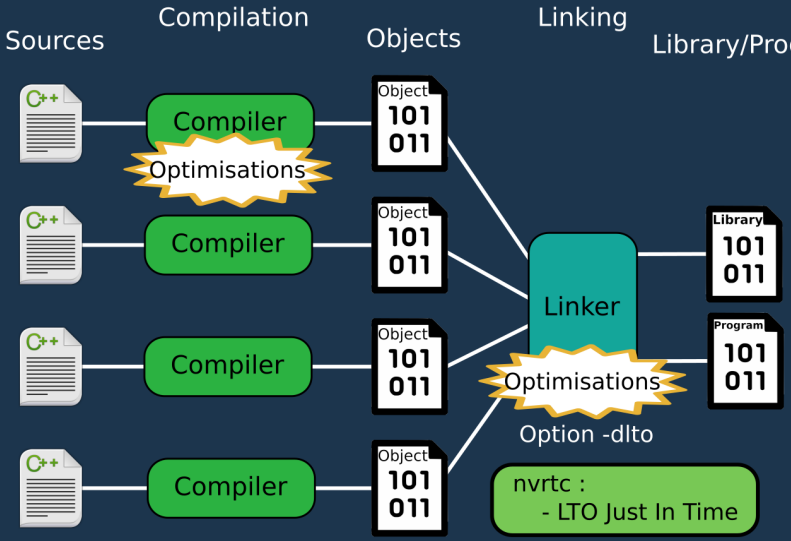
Link Time Optimization



LTO Optimisations :
- Remove dead code (even kernel)

LTO :
- Faster compilation
- Slower linking
- Speed up execution time from 5 to 91 %

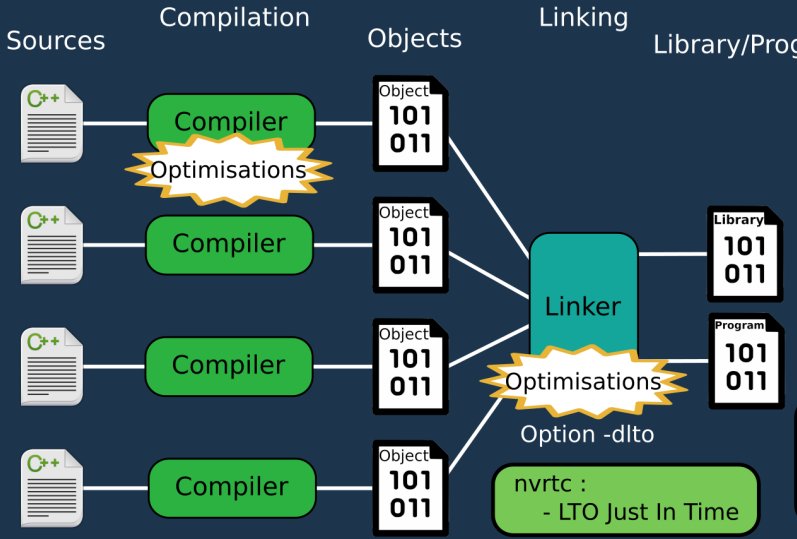
Link Time Optimization



LTO Optimisations :
- Remove dead code (even kernel)

LTO :
- Faster compilation
- Slower linking
- Speed up execution time from 5 to 91 %

Link Time Optimization

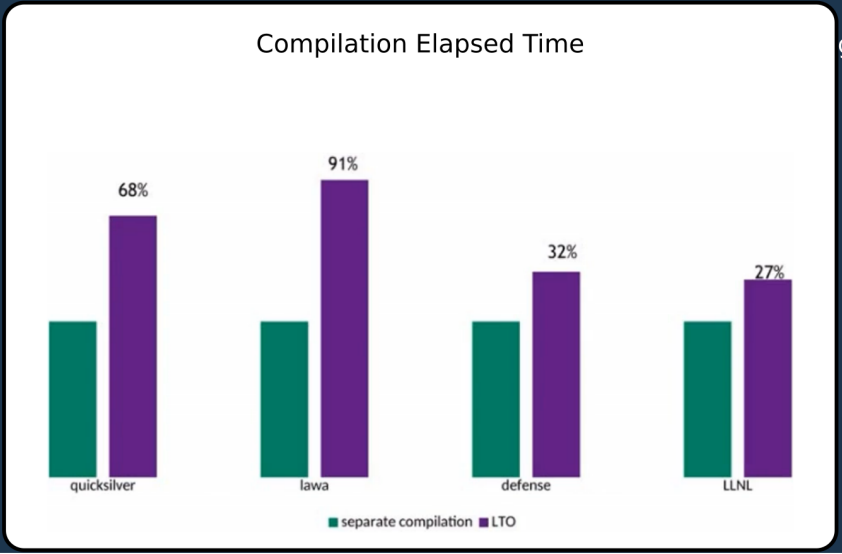


LTO Optimisations :
- Remove dead code (even kernel)

LTO :
- Faster compilation
- Slower linking
- Speed up execution time from 5 to 91 %

Warning :
No compatibility on CUDA minor versions 11.X
Only from CUDA 12.0

Link Time Optimization



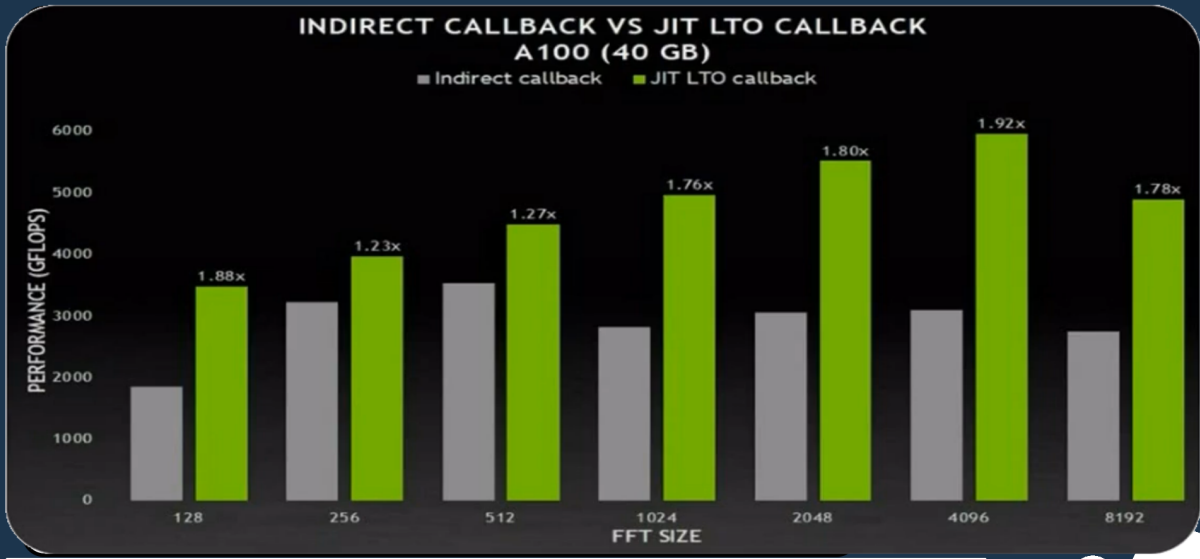
gram

LTO Optimisations :
- Remove dead code (even kernel)

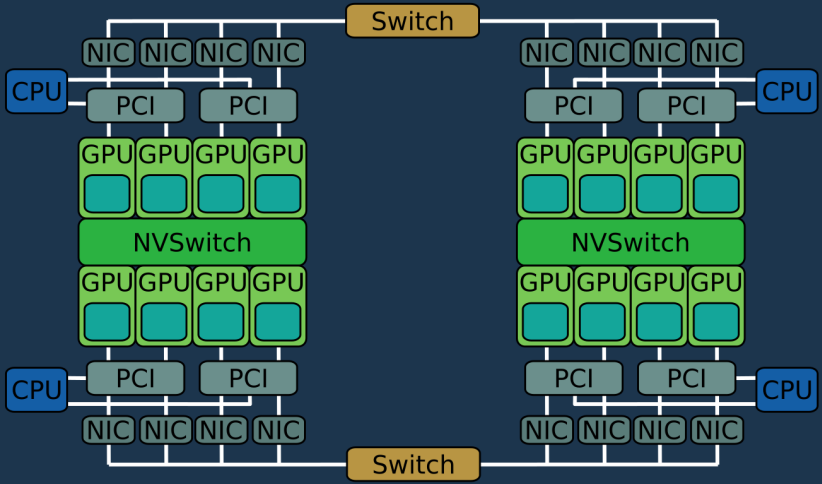
LTO :
- Faster compilation
- Slower linking
- Speed up execution time from 5 to 91 %

Warning :
No compatibility on CUDA minor versions 11.X
Only from CUDA 12.0

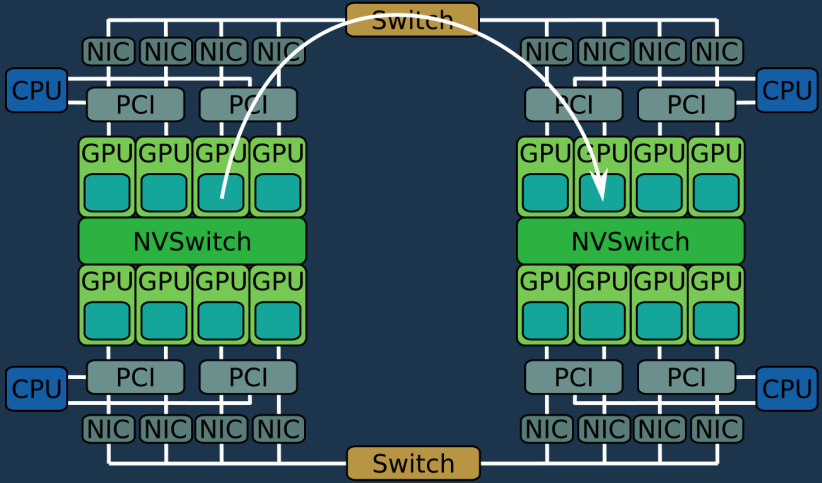
Link Time Optimization



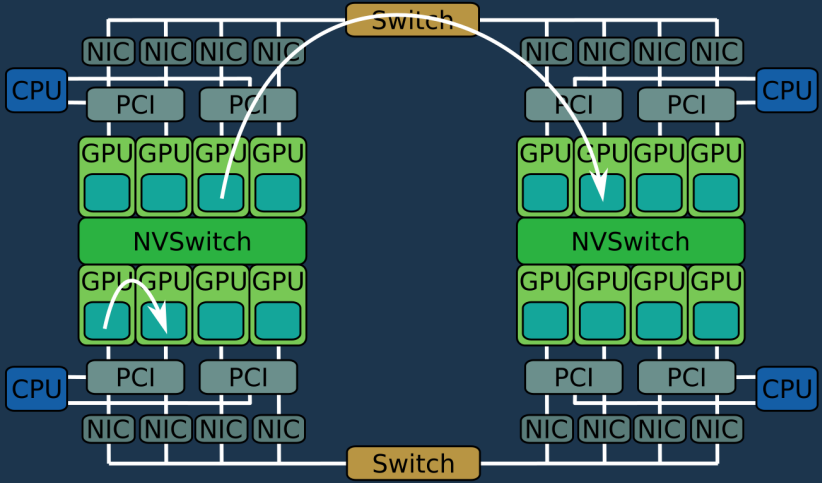
Communication : NVShmem



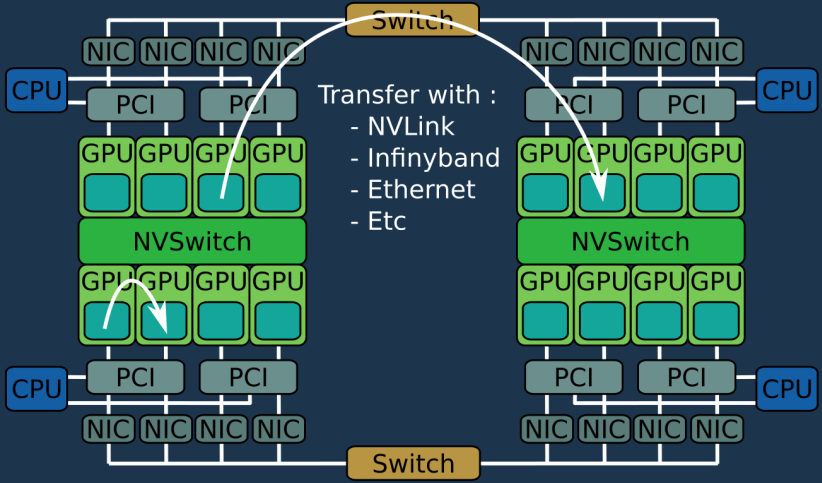
Communication : NVShmem



Communication : NVShmem

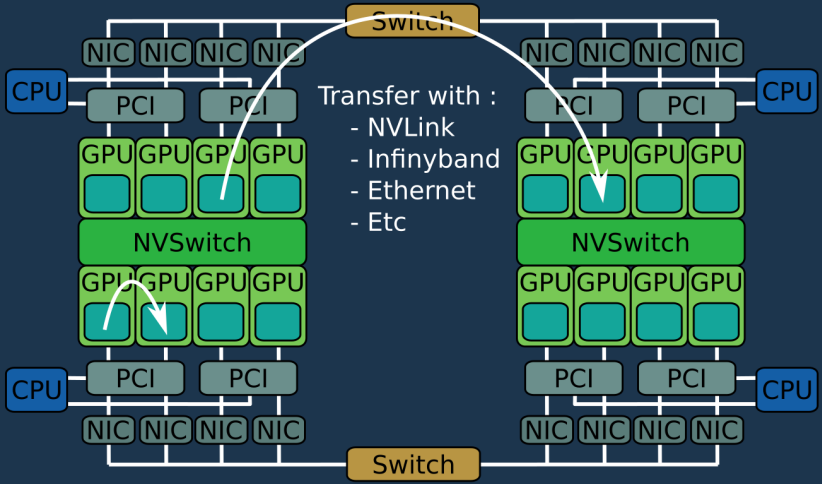


Communication : NVShmem



Communication : NVShmem

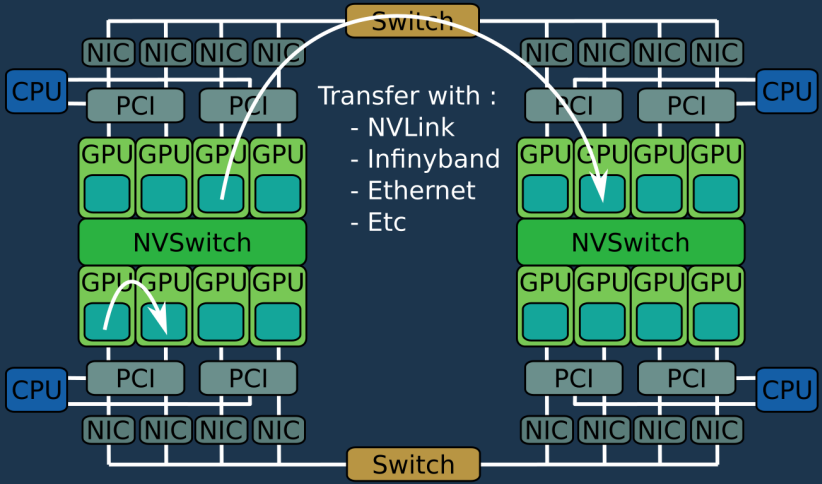
Based on OpenShmem



Communication : NVShmem

Based on OpenShmem

Scalable point to point communication

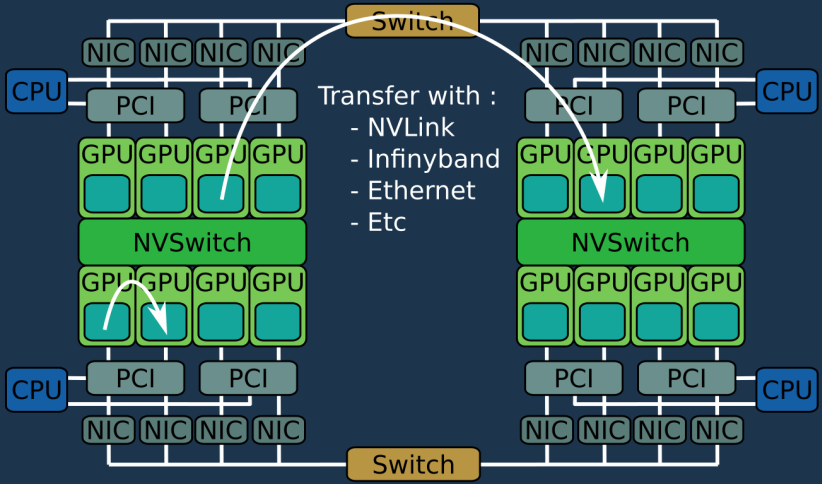


Communication : NVShmem

Based on OpenShmem

Scalable point to point communication

Symetric allocations



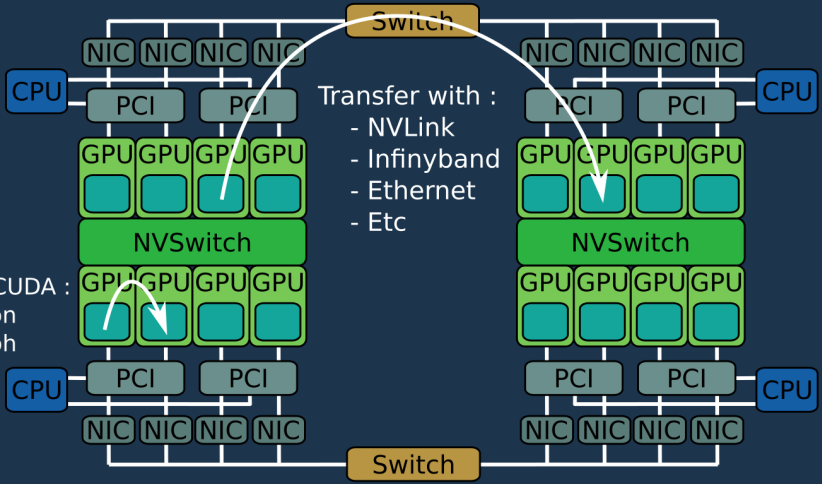
Communication : NVShmem

Based on OpenShmem

Scalable point to point communication

Symetric allocations

Communication integrated with CUDA :
- GPU Kernel-Initiated Operation
- Operation CUDA/Stream graph
- CPU Initiated operations



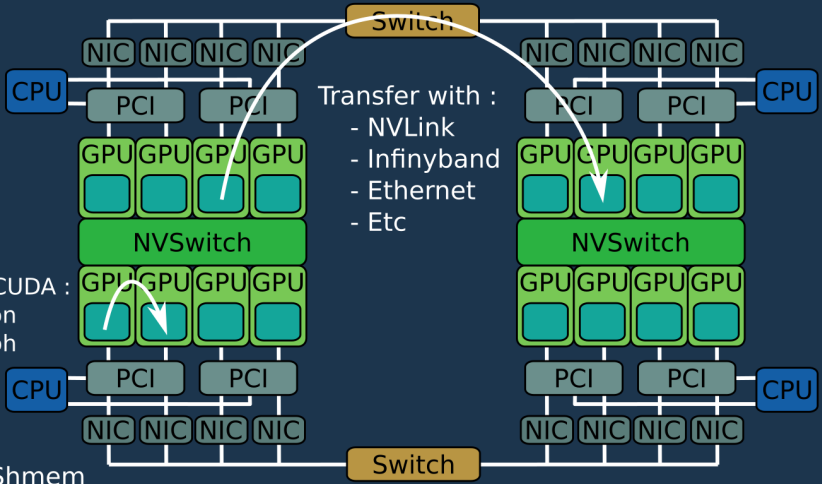
Communication : NVShmem

Based on OpenShmem

Scalable point to point communication

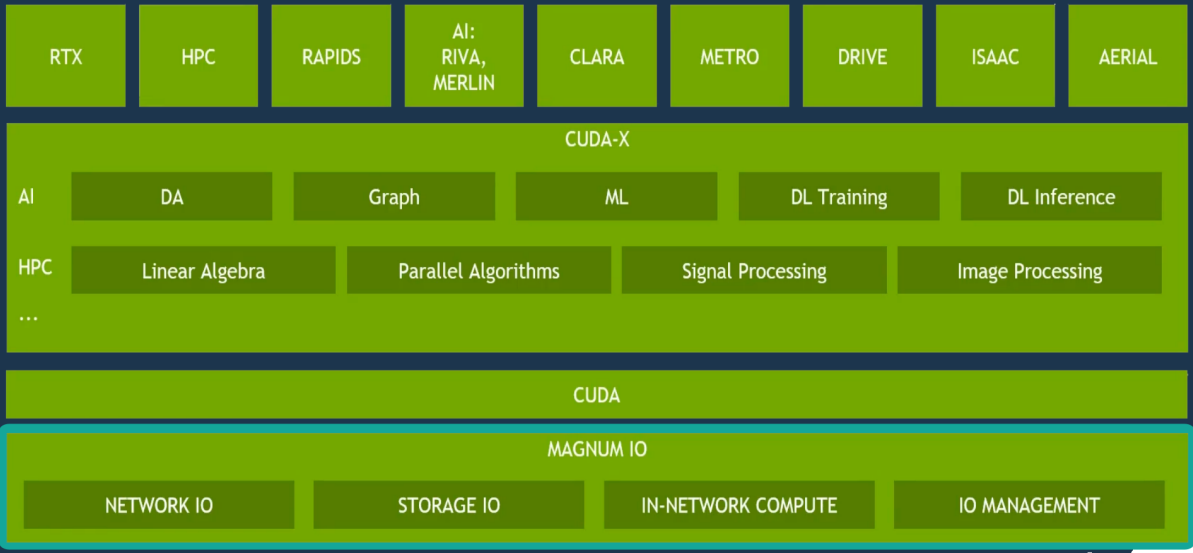
Symetric allocations

Communication integrated with CUDA :
- GPU Kernel-Initiated Operation
- Operation CUDA/Stream graph
- CPU Initiated operations

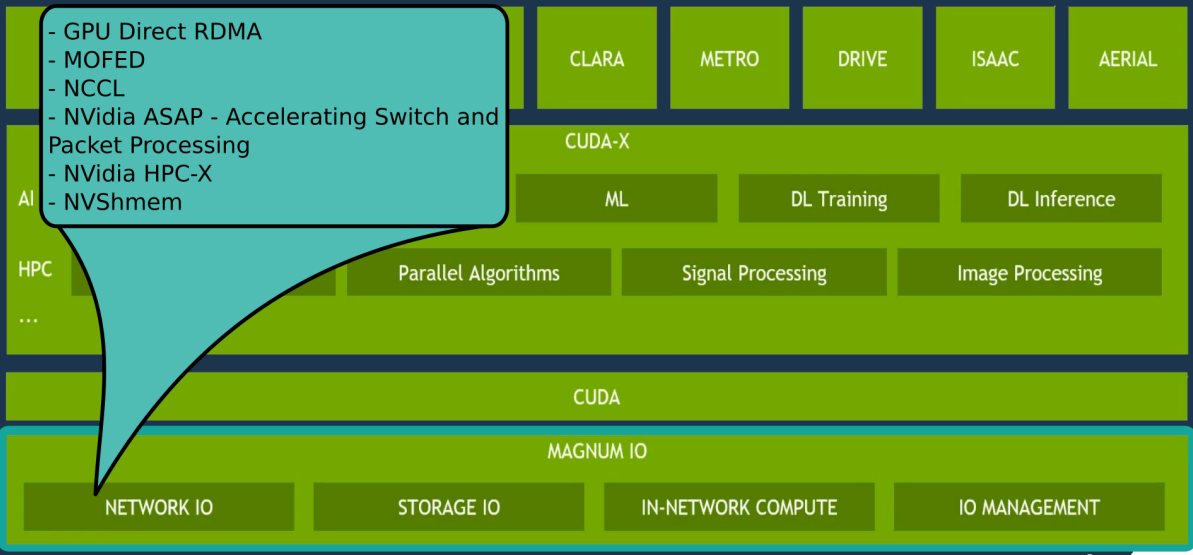


MagnumIO is based on NVShmem

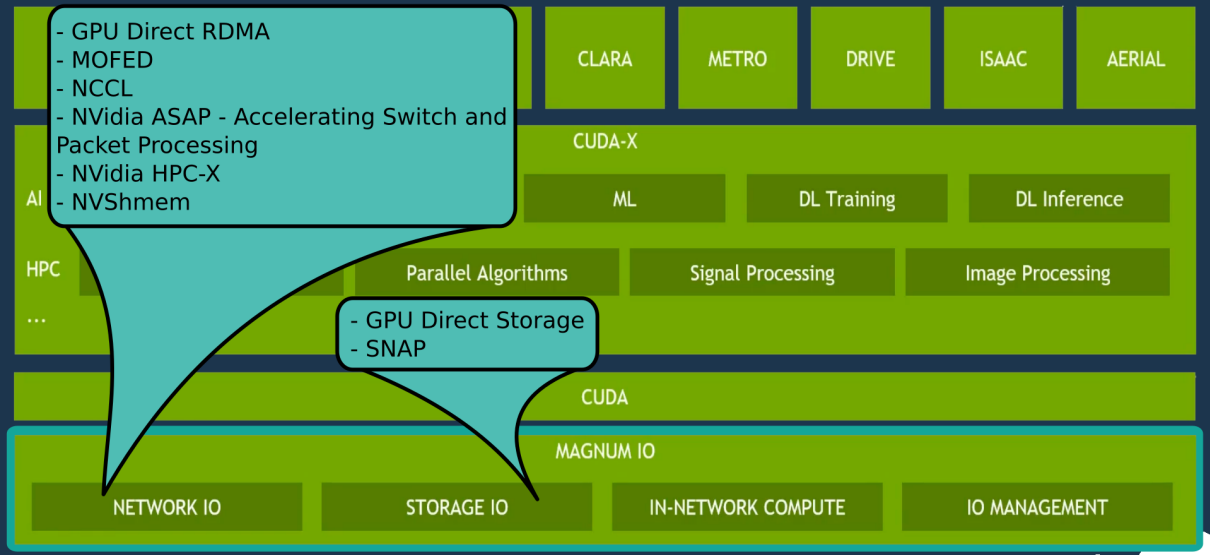
Communication : MagnumIO



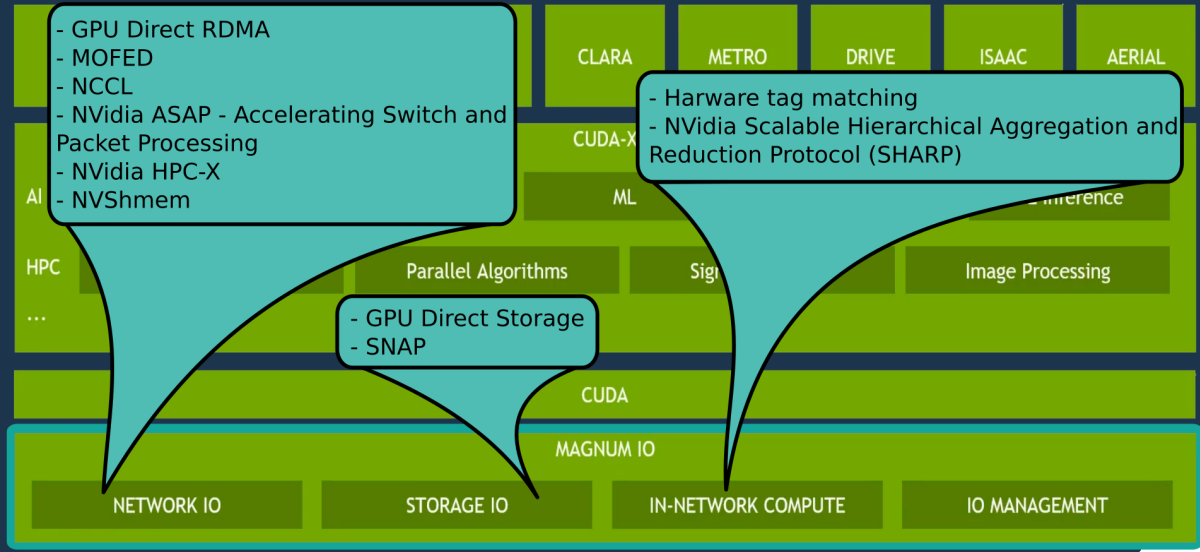
Communication : MagnumIO



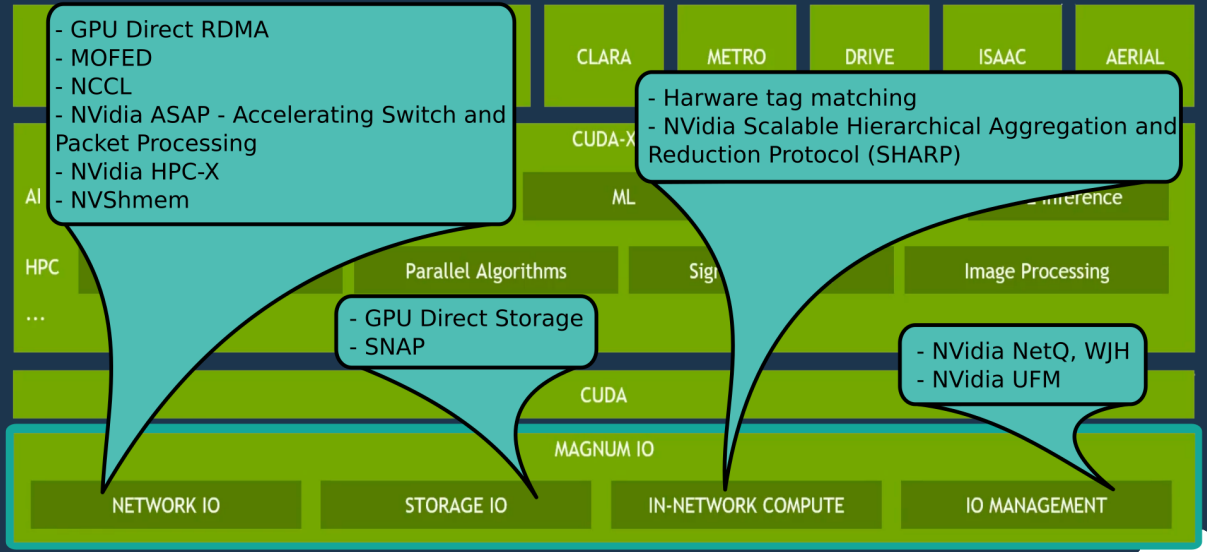
Communication : MagnumIO



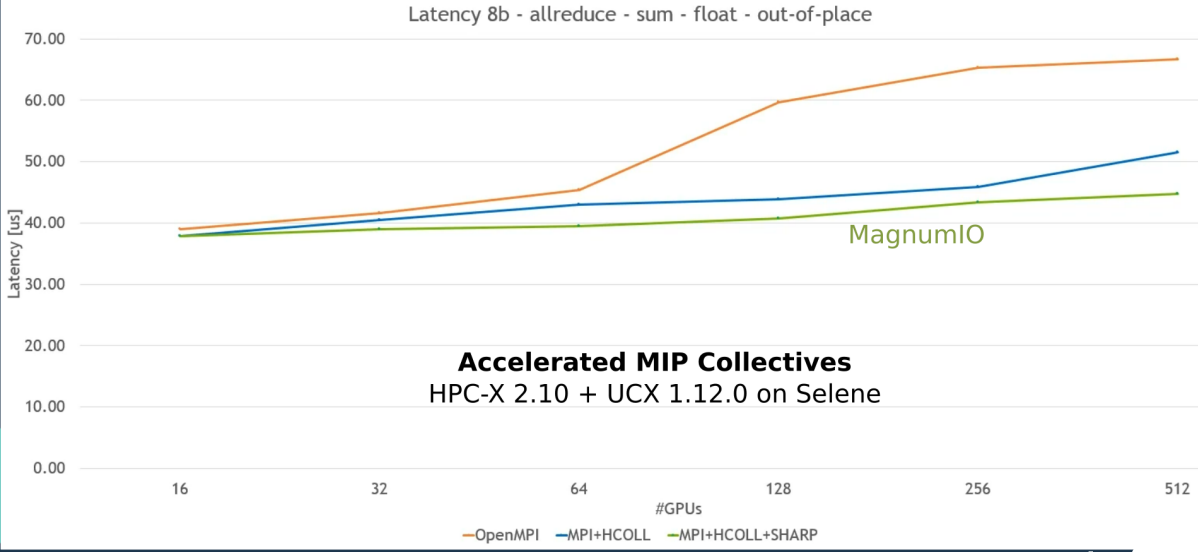
Communication : MagnumIO



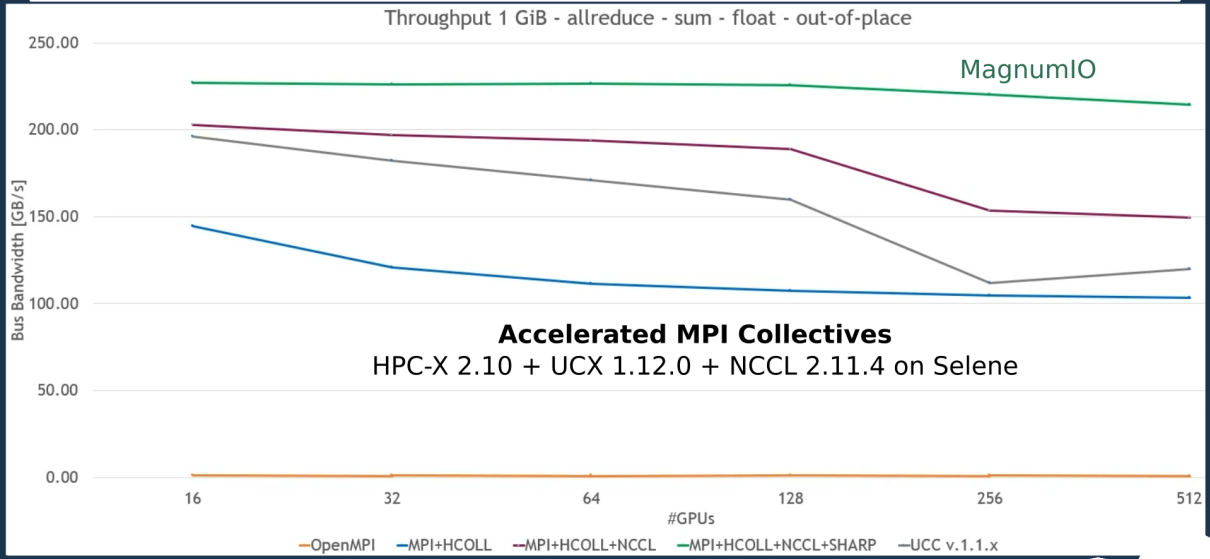
Communication : MagnumIO



Communication : MagnumIO

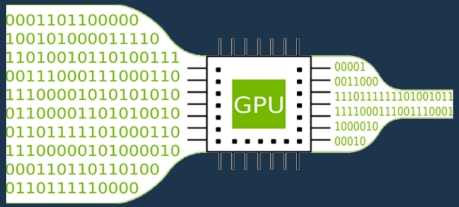


Communication : MagnumIO



Data Compression : NVComp

C/C++ standard calls

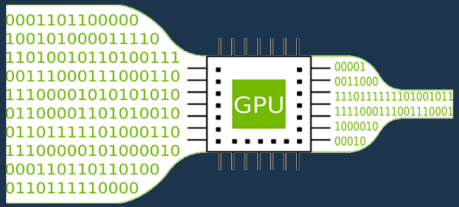


Compression / decompression algorithms :

- LZ4
- Snappy
- DGeflate
- Cascaded
- Bitcomp
- ANS (Asymetric Numeral System)

Data Compression : NVComp

C/C++ standard calls



Release 2.2 :

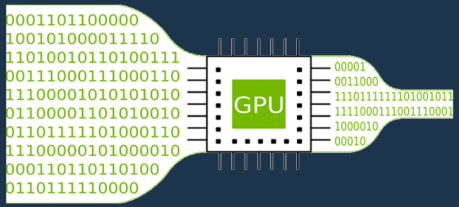
- Can be used with **GPU Direct Storage**
- High Level API simplified
- No kernel call available

Compression / decompression algorithms :

- LZ4
- Snappy
- DGeflate
- Cascaded
- Bitcomp
- ANS (Asymetric Numeral System)

Data Compression : NVComp

C/C++ standard calls



Compression / decompression algorithms :

- LZ4
- Snappy
- DGeflate
- Cascaded
- Bitcomp
- ANS (Asymetric Numeral System)

Release 2.2 :

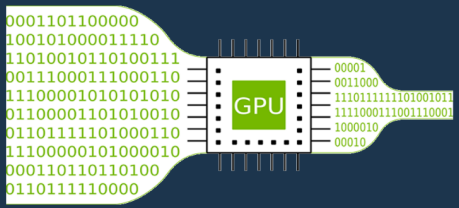
- Can be used with **GPU Direct Storage**
- High Level API simplified
- No kernel call available

Some advices :

- To be used on files of **few MB**
- With **chunks** > 8kB
- Decompressed data must be accessible by the GPU

Data Compression : NVComp

C/C++ standard calls



Compression / decompression algorithms :

- LZ4
- Snappy
- DGeflate
- Cascaded
- Bitcomp
- ANS (Asymetric Numeral System)

Release 2.2 :

- Can be used with **GPU Direct Storage**
- High Level API simplified
- No kernel call available

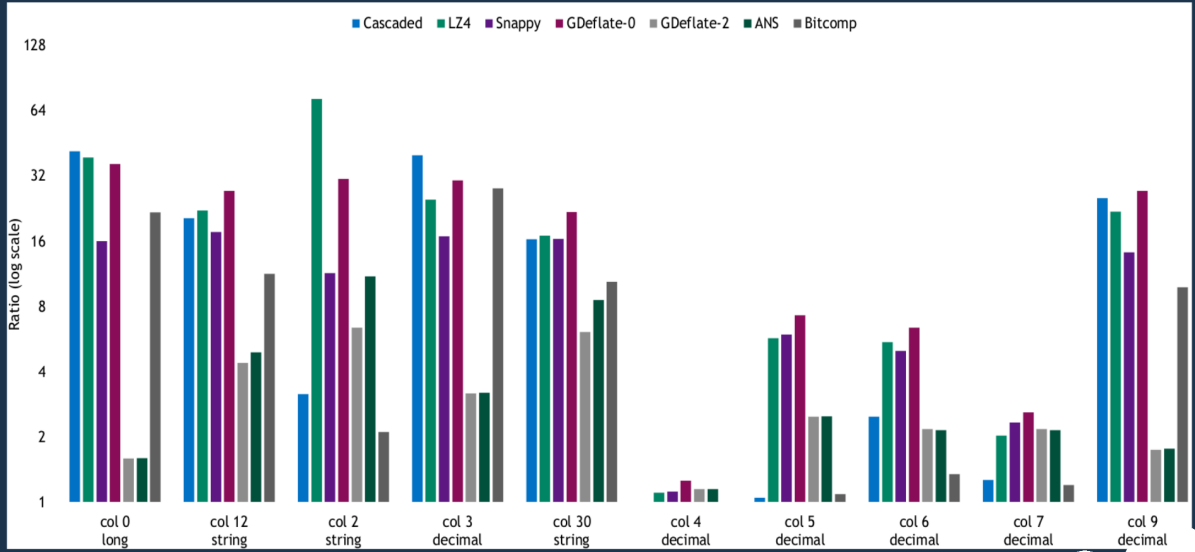
News in future release 2.3 : 

- Add **Zstd** decompression
- Add **BitShuffle** (**float** compression)

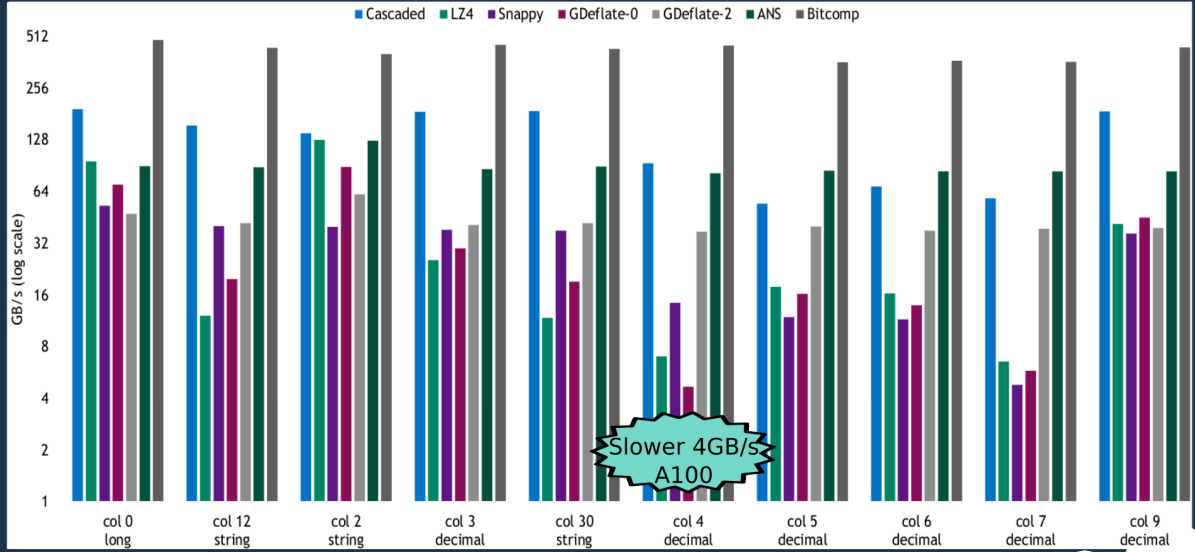
Some advices :

- To be used on files of **few MB**
- With **chunks** > 8kB
- Decompressed data must be accessible by the GPU

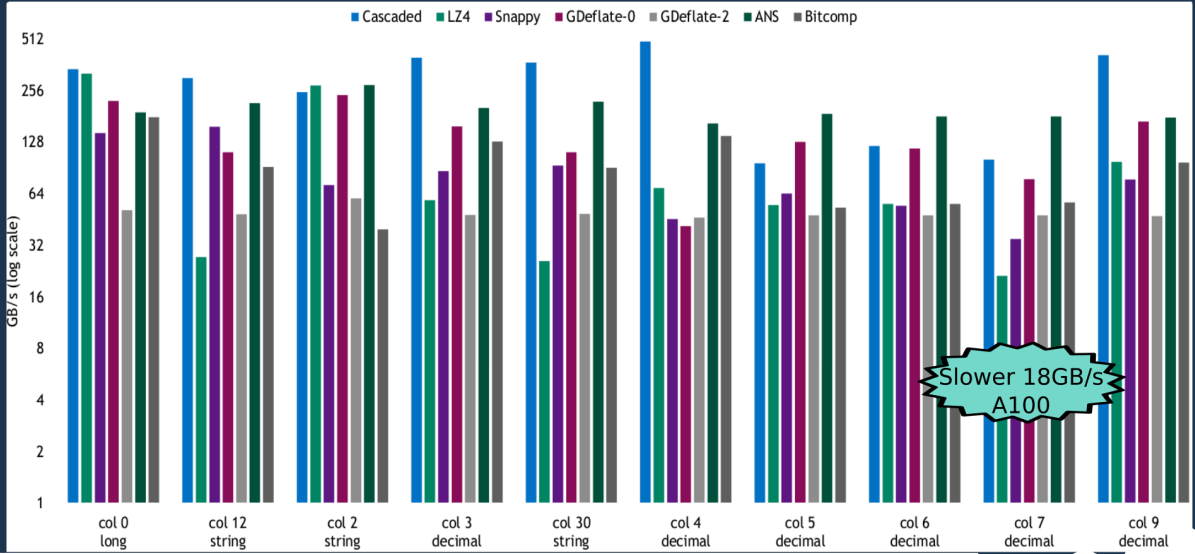
Data Compression : NVComp



Data Compression : NVComp



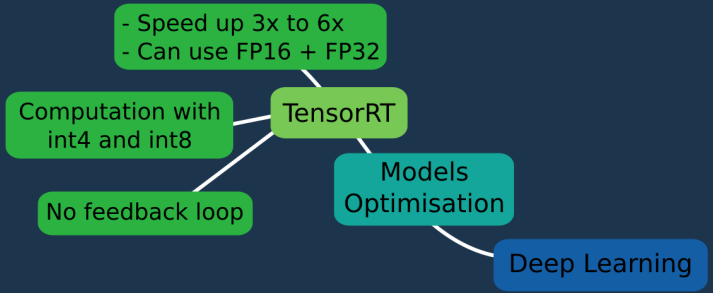
Data Compression : NVComp



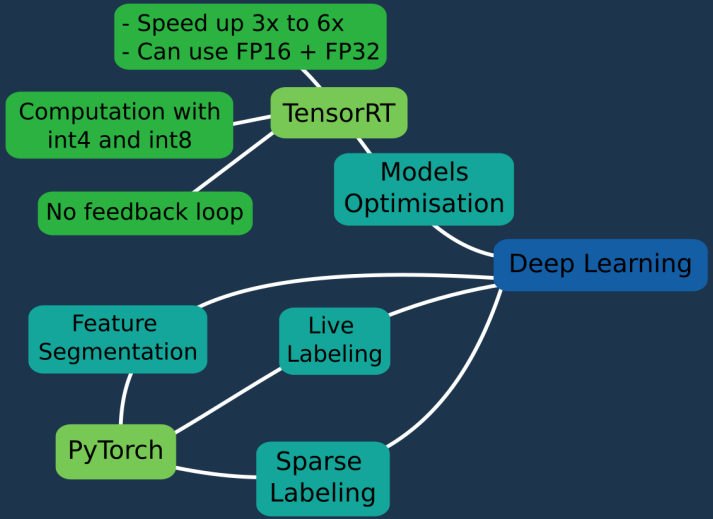
Deep Learning

Deep Learning

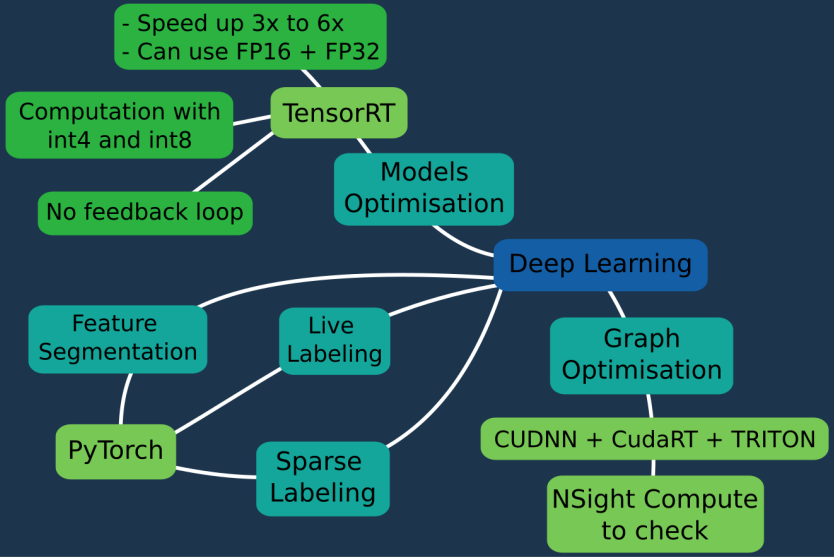
Deep Learning



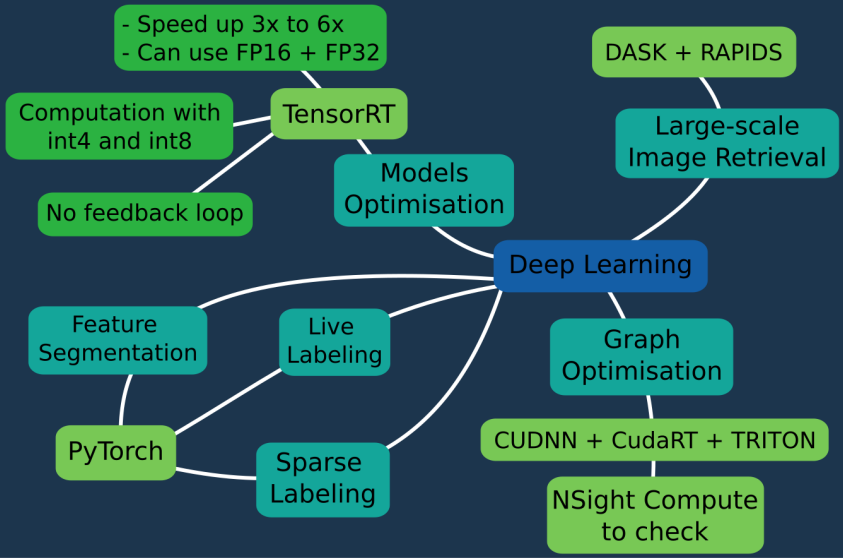
Deep Learning



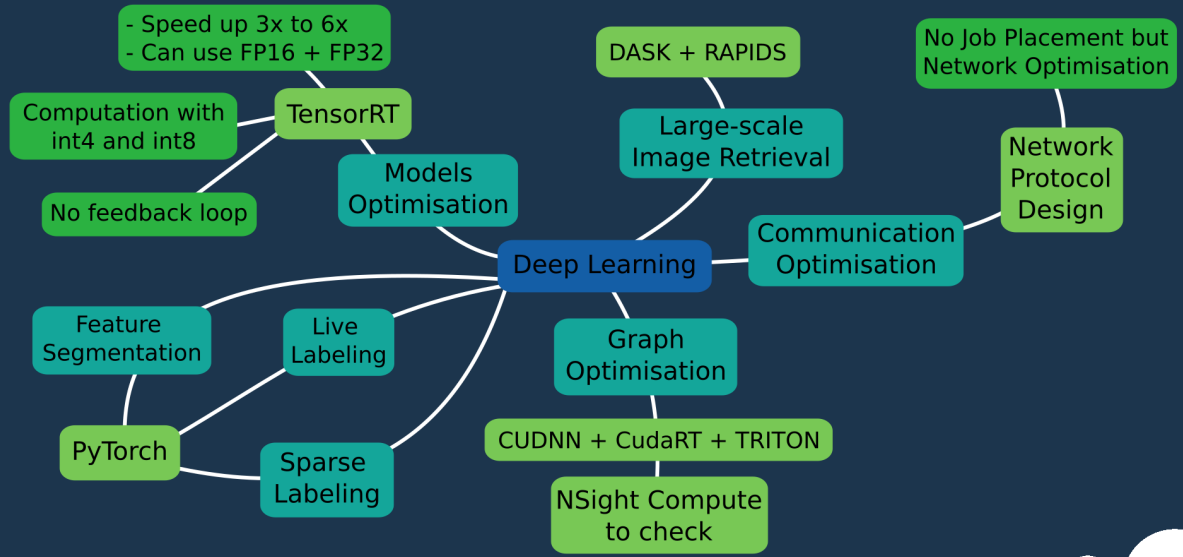
Deep Learning



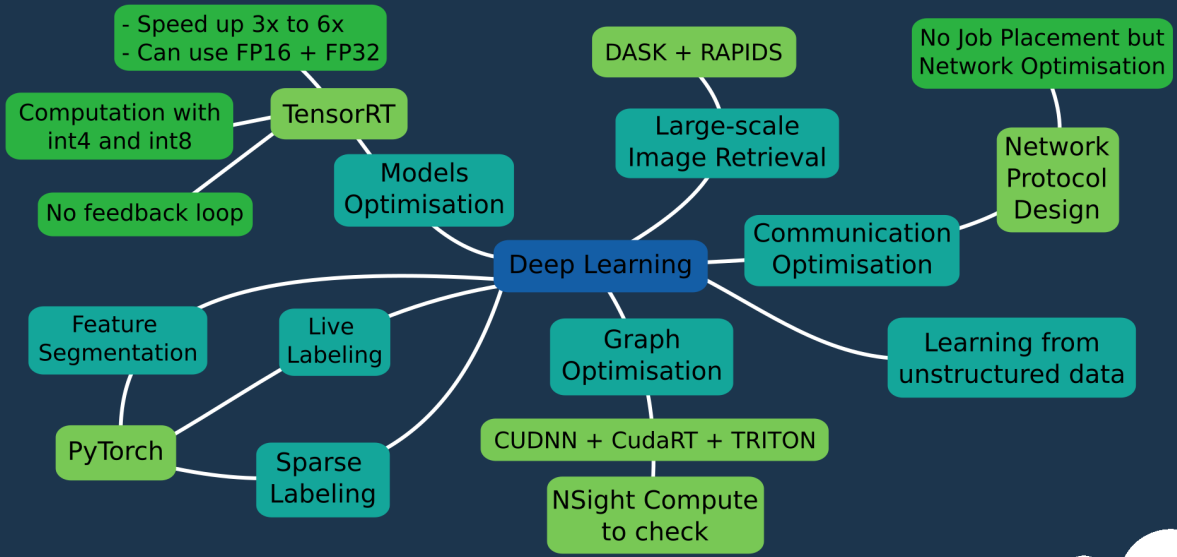
Deep Learning



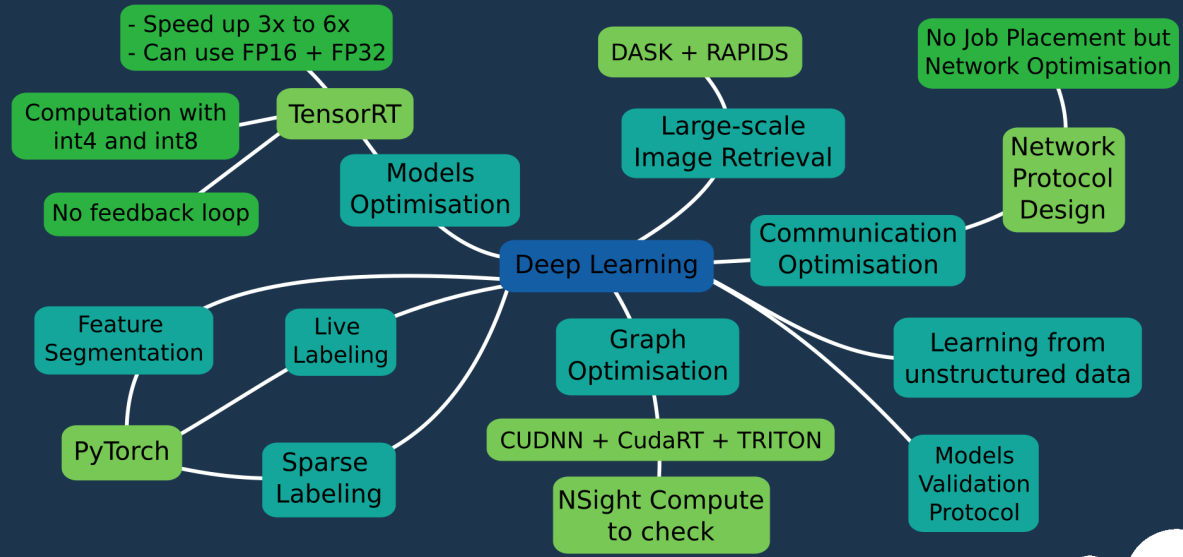
Deep Learning



Deep Learning



Deep Learning



Link to a big summary

GTC 2022 notes from Reprises website

Sébastien Gignoux : l'architecture Fermat (see 11h-11h15 of first day)