FASTER3

A Flexible FPGA/CPU/GPU Architecture for High-Throughput DAQ

Assemat Julian
Beaudou Clément
Carniol Benjamin
Carreau Thomas
Cubéro Guillaume
Etasse David
Fontbonne Cathy
Franck de Preaumont Hugues

Guesnon Sandrine
Houarner Charles
Leterrier Laurent
Matta Adrien
Noury Fabien
Ingouf François
Vandamme Christophe
Yvon Isabelle
Zounda Paul

https://faster.in2p3.fr

Fast Acquisition SysTem for nuclEar Research



Caen, Normandy — ~230 km / 2h from Paris

Laboratoire de Physique Corpusculaire de Caen

UMR 6534 — CNRS/IN2P3 · ENSICAEN · Université de Caen Normandie

1947	~96	36
Founded	Staff	Researchers

Research areas

- Nuclear physics — structure & reactions
- Particle physics & fundamental interactions
- Applications — hadrontherapy, electronuclear cycle

Technical services

IT | Mechanical | Instrumentation | Electronics & Micro-electronics

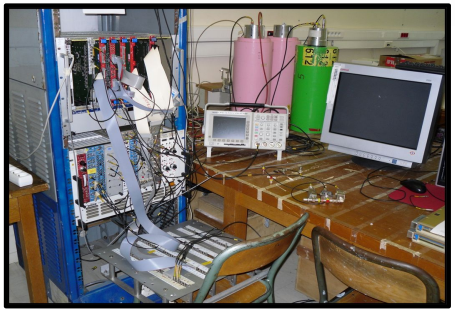
AGENDA

1. FASTER2 – A mature, proven data acquisition system based on FPGA modules, deployed across multiple nuclear physics laboratories.

2. FASTER3 – The next generation of FASTER with more channels, higher bandwidth, lower cost per channel, and flexible FPGA/CPU/GPU processing.
    - Overview
    - ZMQ instead of simple TCP/IP protocol
    - RDMA for datapath
    - Benchmarks

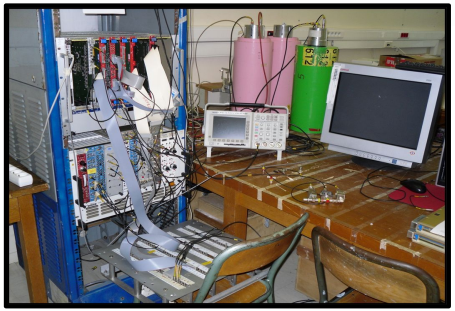
David Etasse → etasse@lpccaen.in2p3.fr, LPC Caen/IN2P3/CNRS

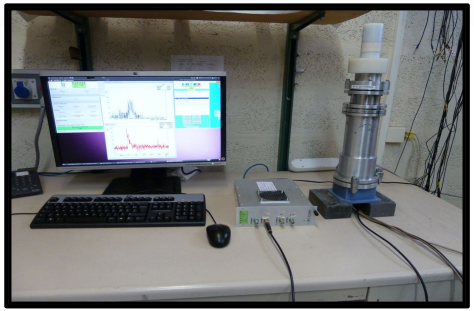
FASTER2 → Historic



2000 ⟶

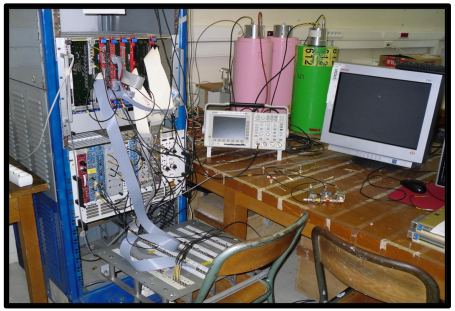
FASTER2 → Historic

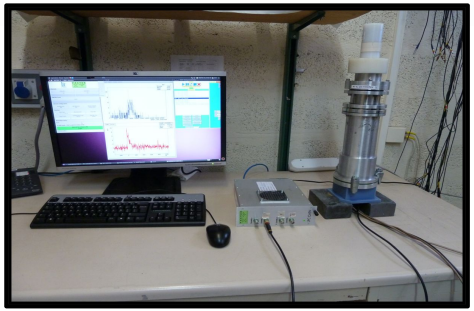




2000 ⟶ 2008 — FASTERv1 ⟶

FASTER2 → Historic











Setup/Control

faster_setup_gui
faster_gui
faster_daq_shell

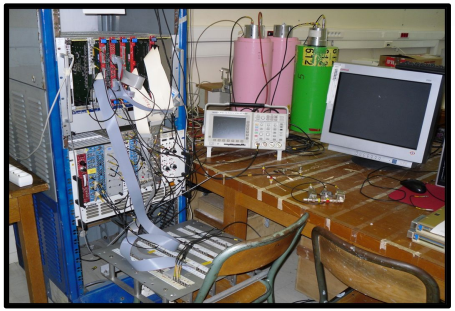


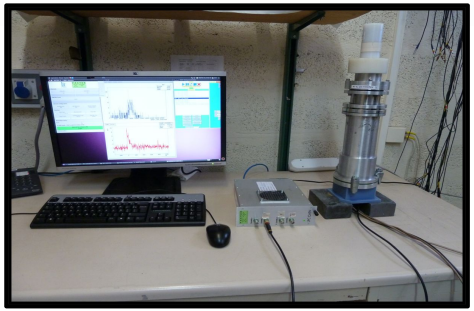
Analysis (Online/Offline)

RHB
nptool
fasterac
pfasterac

2000 → 2008 — FASTERv1 → 2012 — FASTER2 → 2025

FASTER2 → Historic











Setup/Control

faster_setup_gui
faster_gui
faster_daq_shell



Analysis (Online/Offline)

RHB
nptool
fasterac
pfasterac

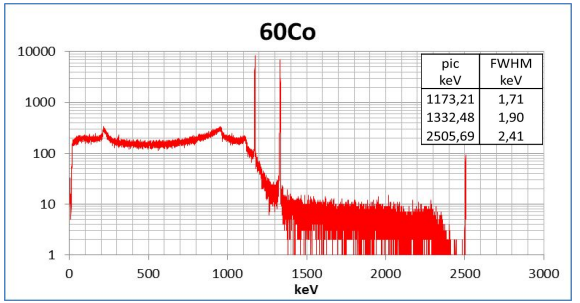
2000 → 2008 FASTERv1 → 2012 FASTER2 → 2025





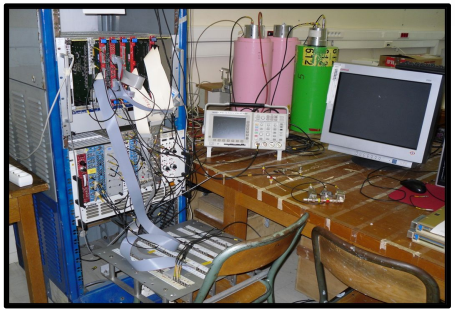
FASTER-CRRC4
FASTER-TRAPEZ-TDC

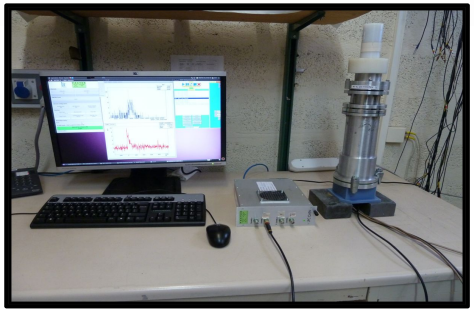




HPGe detector, MOSAHR board, FASTER_CRRC4

FASTER2 → Historic











Setup/Control

faster_setup_gui
faster_gui
faster_daq_shell



Analysis (Online/Offline)

RHB
nptool
fasterac
pfasterac

2000 → 2008 FASTERv1 → 2012 FASTER2 → 2025



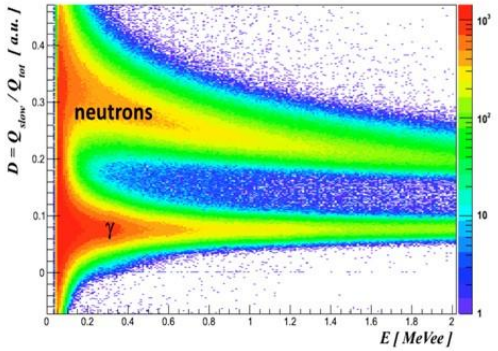




FASTER-CRRC4
FASTER-TRAPEZ-TDC

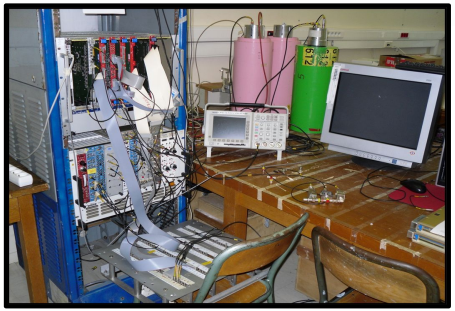


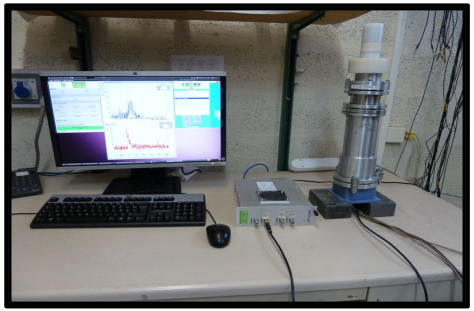
FASTER-QDC-TDC$_{HR}$
FASTER-QT2T
FASTER-RF
FASTER-SCALER
FASTER-SAMPLER



Demon detector, CARAS board, FASTER-QDC-TDC$_{HR}$

FASTER2 → Historic











Setup/Control

faster_setup_gui
faster_gui
faster_daq_shell



Analysis (Online/Offline)

RHB
nptool
fasterac
pfasterac

2000 → 2008 FASTERv1 → 2012 FASTER2 → 2025

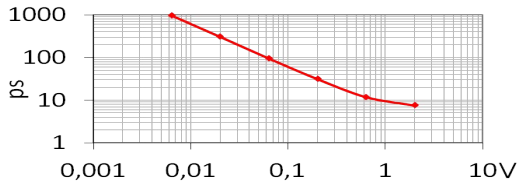






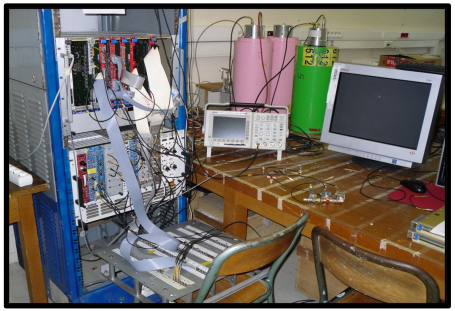
FASTER-CRRC4
FASTER-TRAPEZ-TDC$_{HR-31ps}$

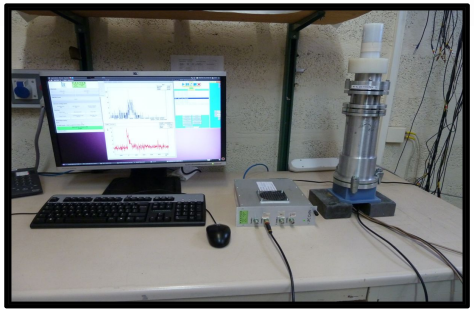
FASTER-QDC-TDC$_{HR-7,8ps}$
FASTER-QT2T
FASTER-RF
FASTER-SCALER
FASTER-SAMPLER



Time resolution
VS
signal amplitude

FASTER2 → Historic











Setup/Control

faster_setup_gui
faster_gui
faster_daq_shell



Analysis (Online/Offline)

RHB
nptool
fasterac
pfasterac

2000 → 2008 FASTERv1 → 2012 FASTER2 → 2025







FASTER-CRRC4
FASTER-TRAPEZ-TDC



FASTER-QDC-TDC$_{HR}$
FASTER-QT2T
FASTER-RF
FASTER-SCALER
FASTER-SAMPLER

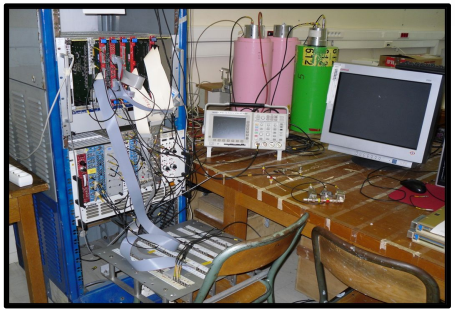


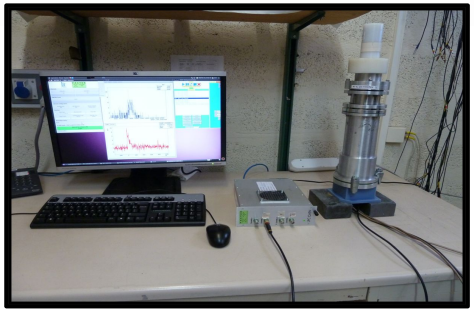
FASTER-ELECTROMETER

FASTER_HV



FASTER2 → Historic











Setup/Control

faster_setup_gui
faster_gui
faster_daq_shell



Analysis (Online/Offline)

RHB
nptool
fasterac
pfasterac

2000 → 2008 FASTERv1 → 2012 FASTER2 → 2025

FASTERv2 features :

- 1 to a few hundred channels synchronized by a 10 MHz clock
- **Triggerless** ~~(Common Dead Time)~~, **Time Stamped** events
- Real Time FPGA processing, No data buffering
- Software triggers (Coincidence, Multiplicity), Multilevel triggers
- 1 to a few million events/second ($\propto$ CPU)
- Control/Setup/Data → Proprietary UDP protocol
- https://faster.in2p3.fr
- https://nptool.in2p3.fr
- https://gitlab.in2p3.fr/cfontbonne/pfasterac
- sudo apt install fasterv2, rhb, fasterac

AGENDA

1. FASTER2 – A mature, proven data acquisition system based on FPGA modules, deployed across multiple nuclear physics laboratories.

2. FASTER3 – The next generation of FASTER with more channels, higher bandwidth, lower cost per channel, and flexible FPGA/CPU/GPU processing.
    - Overview
    - ZMQ instead of simple TCP/IP protocol
    - RDMA for datapath
    - Benchmarks

David Etasse → etasse@lpccaen.in2p3.fr, LPC Caen/IN2P3/CNRS

FASTER3

Data acquisition,
Slow control,
Synchronization (IJCLab WR)

daniel.charlet@ijclab.in2p3.fr
antoine.back@ijclab.in2p3.fr



Board Design, Firmware,
VHDL Algorithms, Synchronization

→ 5 LPC Caen, 1 CDD $^{2\ years}$, X years,
FASTER2 funding

FASTER3

Data acquisition,
Slow control,
Synchronization (IJCLab WR)

daniel.charlet@ijclab.in2p3.fr
antoine.back@ijclab.in2p3.fr



Board Design, Firmware,
VHDL Algorithms, Synchronization

→ 5 LPC Caen, 1 CDD $^{2\ years}$, X years,
FASTER2 funding

User Algorithms

ML or Traditional
Algorithms

→ LPC Caen or
other laboratories

FASTER3

High-throughput platform, Embedded ML processing, Centralized deployment

SINAPSE
2025 → 2027

SImulateur **N**umérique pour l'**A**pprentissage **P**rofond sur **S**ystème **E**mbarqué
Digital SImulator for Deep Learning on Embedded Systems
→ 4 LPC Caen, 1 CDD [2 years], 2 years, 150K€

Data acquisition, Slow control, Synchronization (IJCLab WR)

daniel.charlet@ijclab.in2p3.fr
antoine.back@ijclab.in2p3.fr



Board Design, Firmware, VHDL Algorithms, Synchronization

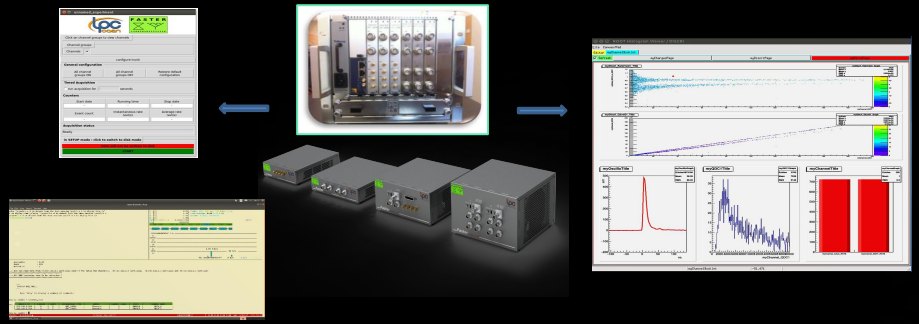
→ 5 LPC Caen, 1 CDD [2 years], X years, FASTER2 funding

User Algorithms

ML or Traditional Algorithms

→ LPC Caen or other laboratories

FASTER3 → FASTER2 vs FASTER3

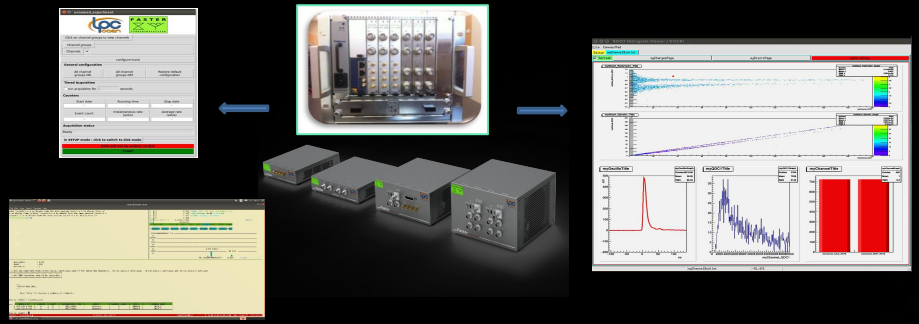




1. Turn on the detectors ↔ Plug in the detectors
   Turn on the FASTER2 unit

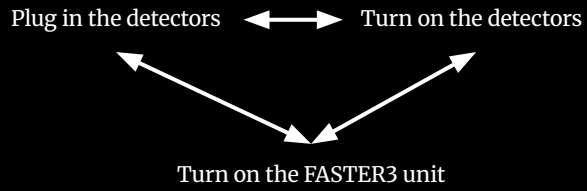
1. Read the Quick Start guide
2. Go to **http://faster.in2p3.fr** (First steps)
3. Turn on the computer
4. Install Ubuntu 24.04 LTS 64-bit
5. Install fasterv2 → **sudo apt install fasterv2**
6. Configure network card
7. Create a directory
8. Use **faster_setup_gui** to configure a FASTER environment
9. Use **faster_gui** or **faster_daq_shell**
10. Launch **RHB** for visualization
11. Start data acquisition

FASTER3 → FASTER2 vs FASTER3





1. Turn on the detectors ↔ Plug in the detectors
   Turn on the FASTER2 unit
1. Read the Quick Start guide
2. Go to **http://faster.in2p3.fr** (First steps)
3. Turn on the computer
4. Install Ubuntu 24.04 LTS 64-bit
5. Install fasterv2 → **sudo apt install fasterv2**
6. Configure network card
7. Create a directory
8. Use **faster_setup_gui** to configure a FASTER environment
9. Use **faster_gui** or **faster_daq_shell**
10. Launch **RHB** for visualization
11. Start data acquisition



FASTER3 → A synchronized multi-channel Oscilloscope module



Desktop modules



FASTER3 → Synchronized multi-channel Oscilloscope Platform

Desktop modules

19” Server

2U 4U

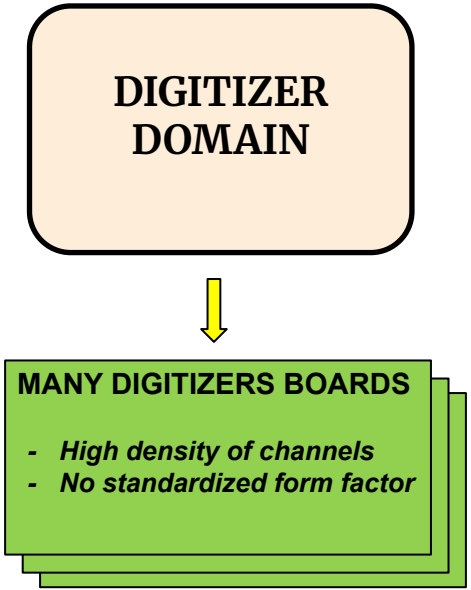


Use prototype daughter boards
4 → 8 Channels

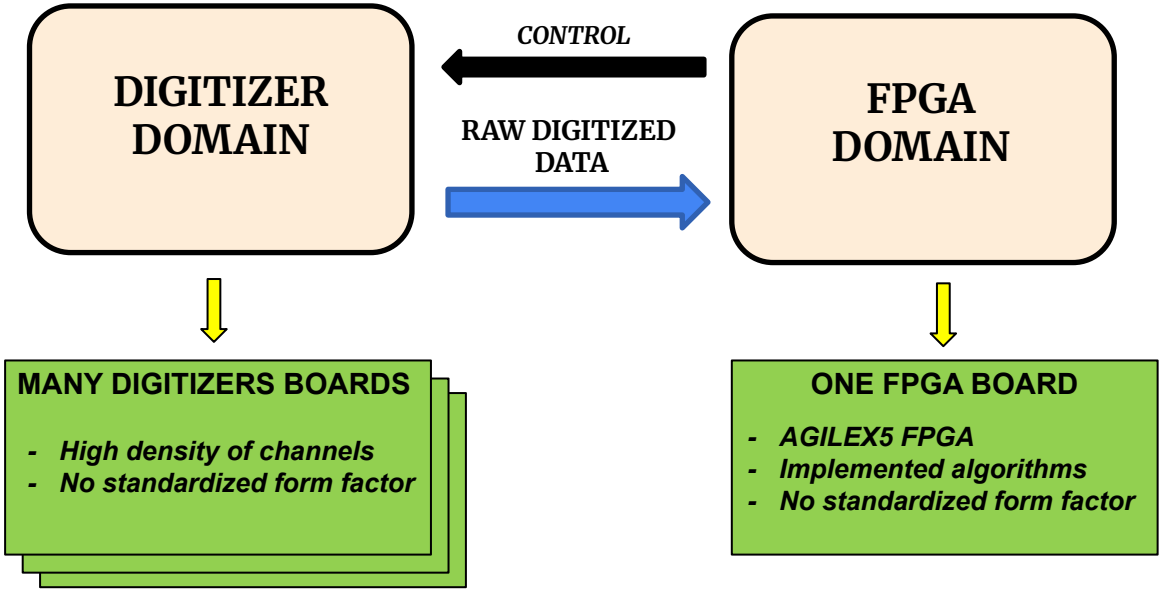
Use production daughter boards
8 → 16 Channels

FASTER3 → OVERALL HARDWARE ARCHITECTURE

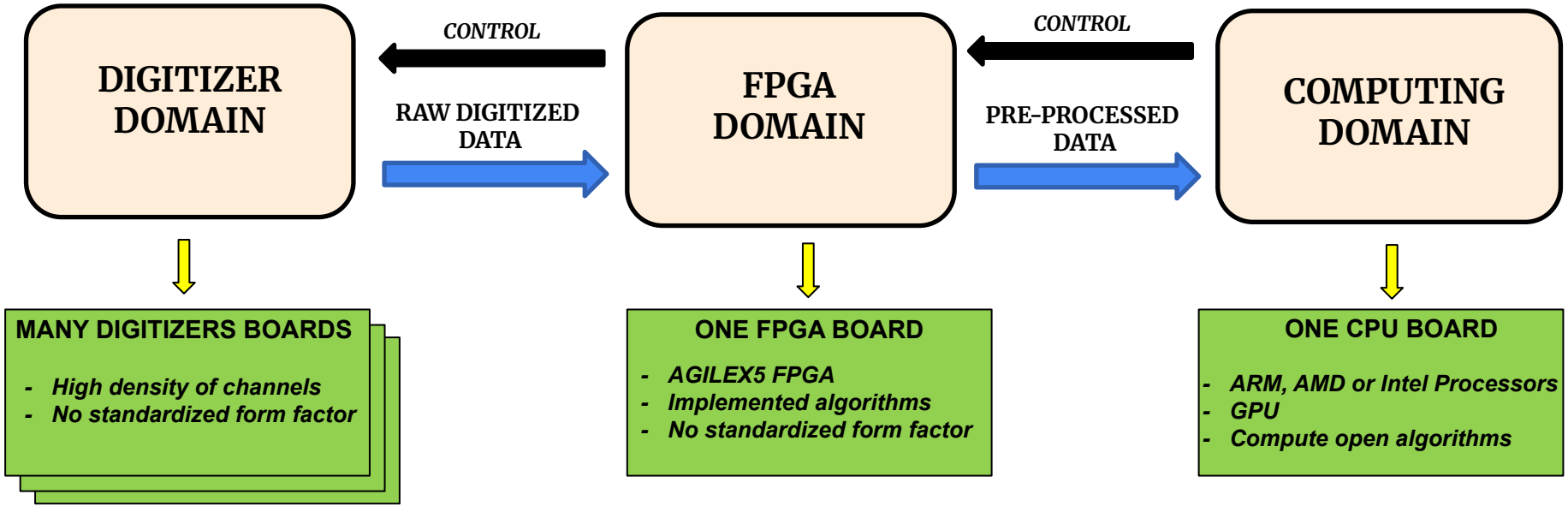
FASTER3 → OVERALL HARDWARE ARCHITECTURE



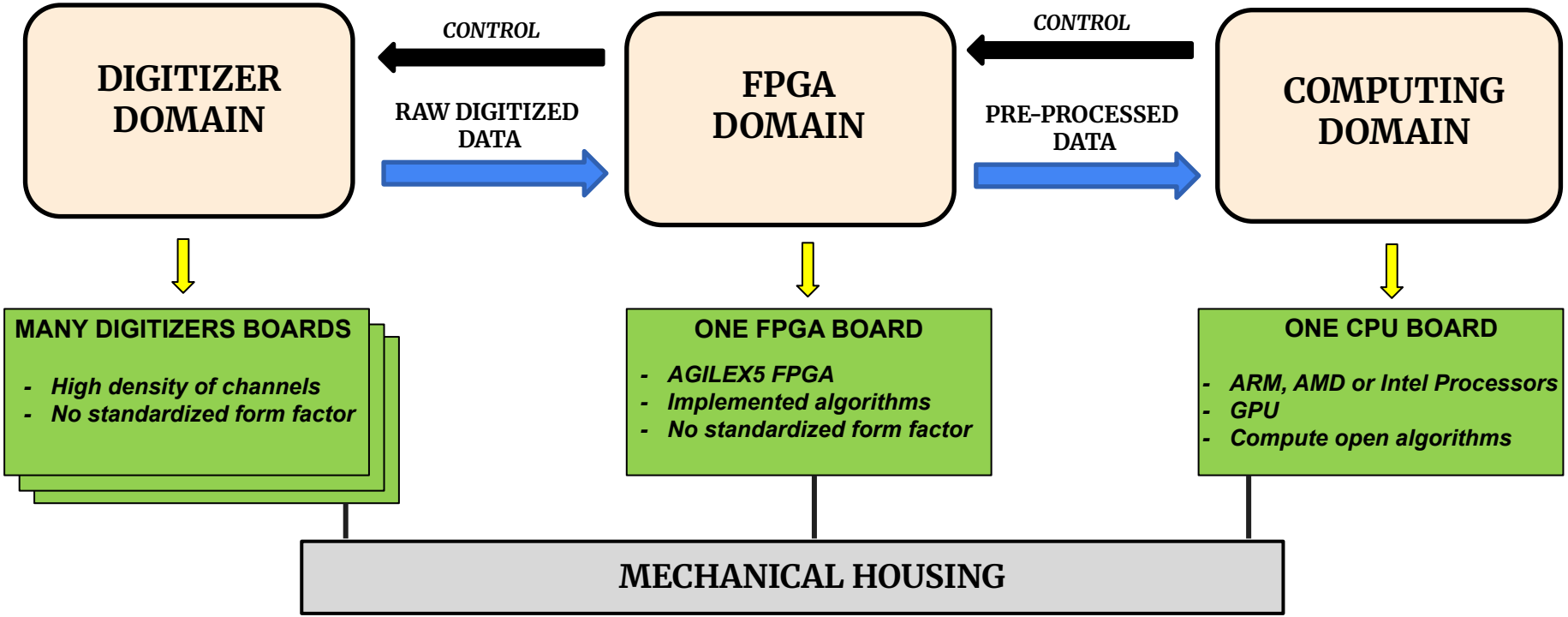
FASTER3 → OVERALL HARDWARE ARCHITECTURE



FASTER3 → OVERALL HARDWARE ARCHITECTURE



FASTER3 → OVERALL HARDWARE ARCHITECTURE



Capabilities :

	FASTER2
Digitizer bitrate	6Gbps
Output bitrate	1Gbps (Ethernet)
Volatile memory (capacity @ bitrate)	1Gb @ 12,8Gbps (DDR3)
Synchronization	10 Mhz clock and T0

Capabilities :

	FASTER2	FASTER3
Digitizer bitrate	6Gbps	192Gbps
Output bitrate	1Gbps (Ethernet)	128Gbps (2*4 lanes PCIe Gen4)
Volatile memory (capacity @ bitrate)	1Gb @ 12,8Gbps (DDR3)	64Gb @ 410Gbps (2*LPDDR5)
Synchronization	10 Mhz clock and T0	White Rabbit or SMART Protocol

Electronic parameters by digitizer board :



FASTER2

GOAL → RESOLUTION	
	MOSHAR
Channels	4
Sampling rate	14bits@125MSps
ENOB target	12
Bandwidth	25MHz

Electronic parameters by digitizer board :



FASTER2 → FASTER3

	GOAL → RESOLUTION	
	MOSHAR	HYDRA
Channels	4	16 (8 for prototype)
Sampling rate	14bits@125MSps	16bits@125MSps
ENOB target	12	12,5
Bandwidth	25MHz	25MHz

Electronic parameters by digitizer board :



FASTER2

GOAL → TIMING	
	CARAS
Channels	2
Sampling rate	12bits@500MSps
ENOB target	10
Bandwidth	100MHz

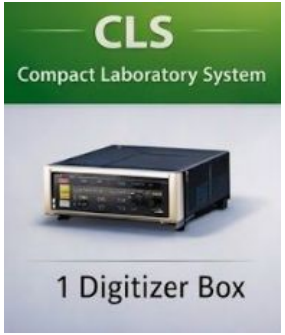
Electronic parameters by digitizer board :



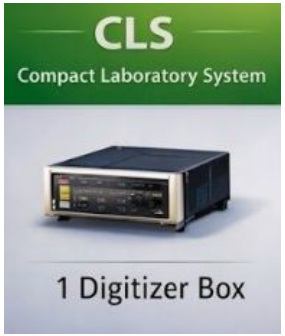
FASTER2 → FASTER3

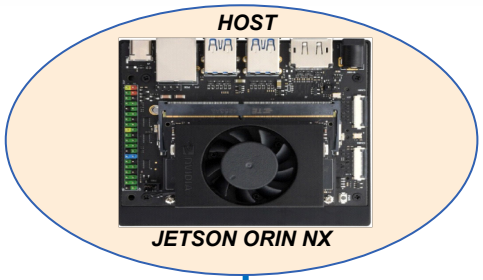
	GOAL → TIMING		
	CARAS	PEGASUS	VIRGO
Channels	2	16 (4 for prototype)	8 (4 for prototype)
Sampling rate	12bits@500MSps	16bits@1,5GSps	12bits@4GSps
ENOB target	10	11	9
Bandwidth	100MHz	300MHz	300MHz

FASTER3 → Compact Laboratory System



FASTER3 → Compact Laboratory System





- 4 to 8 channels
- 16GB,
- 8 Cores CPU,
- 1024 Cores GPU

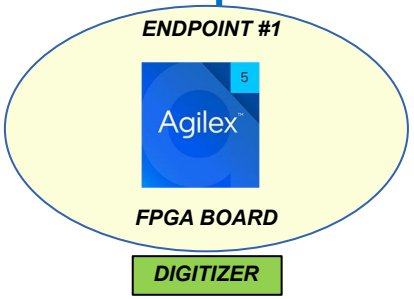
64 Gbps

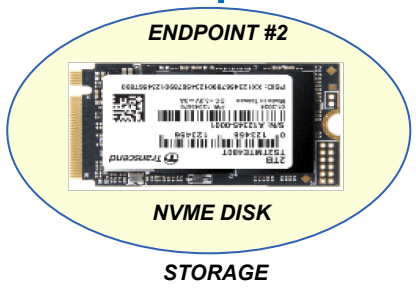
PCIE SWITCH

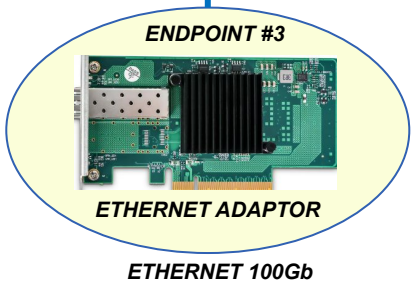
128 Gbps

64 Gbps

128 Gbps







FASTER3 → First CLS module → Mid 2027

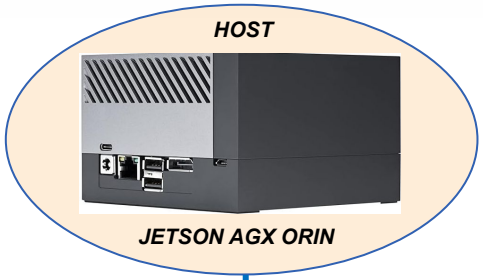


1. 8 Channels 125Msps@16bits
1. Trigger In/Out, Sync In/Out
1. Oled screen (IP, DAQ Status …)
1. Jetson Orin NX (16GB, 8 Cores CPU, 1024 Cores GPU)
1. Ubuntu 24.04 LTS@64 bits
1. Rear Multi-connection (USB, DP, RJ45 1Gbe)
1. Optional 100Gbe PCIe board



FASTER3 → Mid-scale Laboratory System





- 8 to 16 channels
- 64GB,
- 12 Cores CPU,
- 2048 Cores GPU

256 Gbps

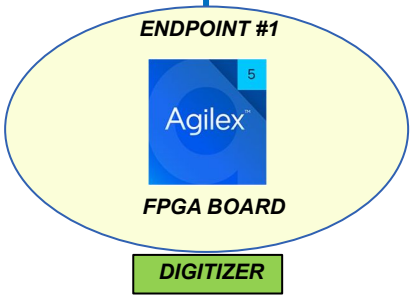
PCIE SWITCH

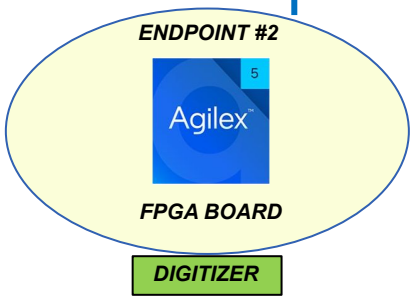
128 Gbps

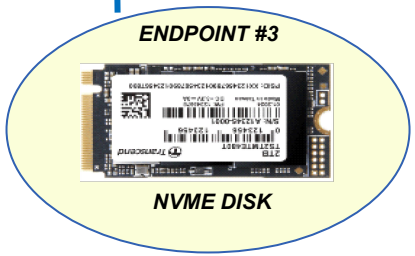
128 Gbps

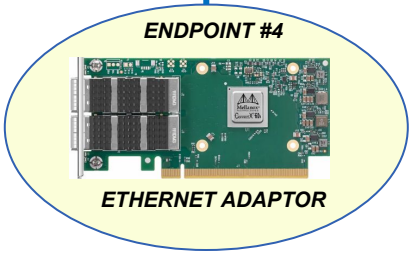
64 Gbps

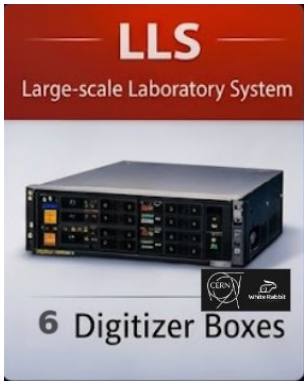
256 Gbps





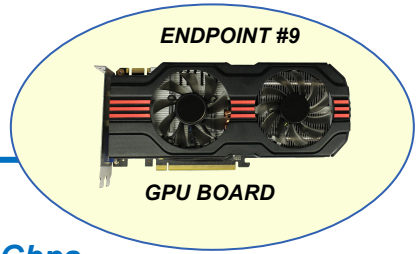






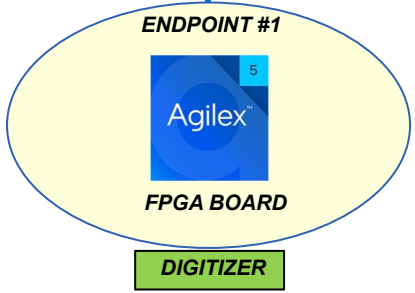
- 48 to 96 channels
- From 12 to 86 cores (24 to 172) threads.
- 16GB to 1TB
- 80 PCIe 5.0 lanes for maximum high-speed connectivity.





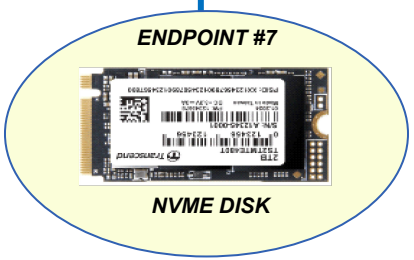
6 x

128 Gbps

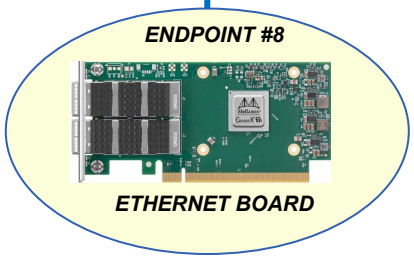


128 Gbps

n x



512 Gbps



FASTER3

High-throughput platform, Embedded ML processing, Centralized deployment

SINAPSE
2025 → 2027

SImulateur **N**umérique pour l'**A**pprentissage **P**rofond sur **S**ystème **E**mbarqué
Digital SImulator for Deep Learning on Embedded Systems
→ 4 LPC Caen, 1 CDD [2 years], 2 years, 150K€

Data acquisition, Slow control, Synchronization (IJCLab WR)

daniel.charlet@ijclab.in2p3.fr
antoine.back@ijclab.in2p3.fr



Board Design, Firmware, VHDL Algorithms, Synchronization

→ 5 LPC Caen, 1 CDD [2 years], X years, FASTER2 funding

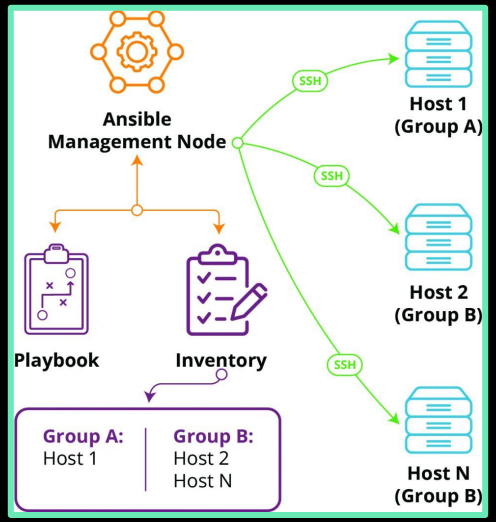
User Algorithms

ML or Traditional Algorithms

→ LPC Caen or other laboratories

FASTER3 → Ansible

- **Ansible** is an open source, command-line IT automation software application written in Python.
- It can configure systems, deploy software, and orchestrate advanced workflows to support application deployment, system updates, and more.



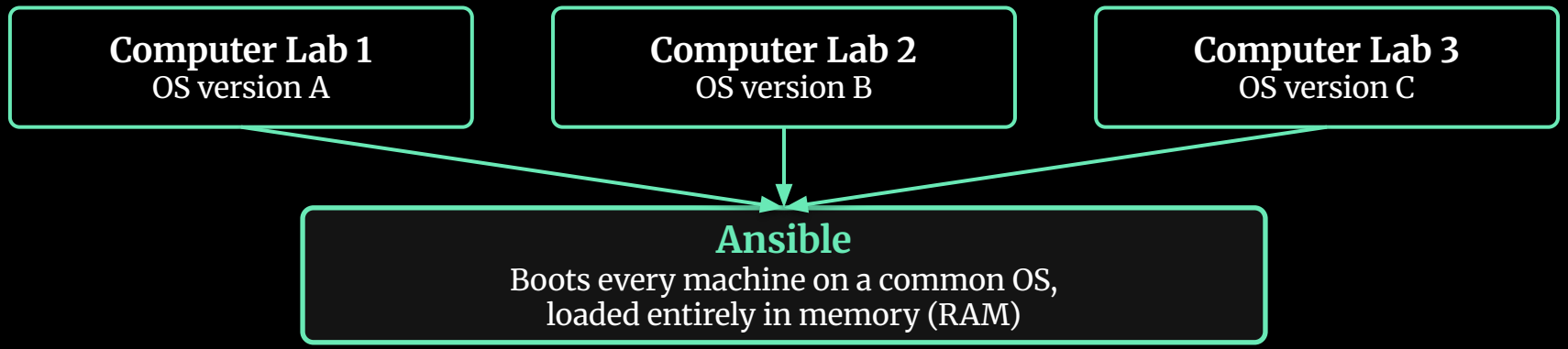
FASTER3 → Ansible Use case

Computer Lab 1
OS version A

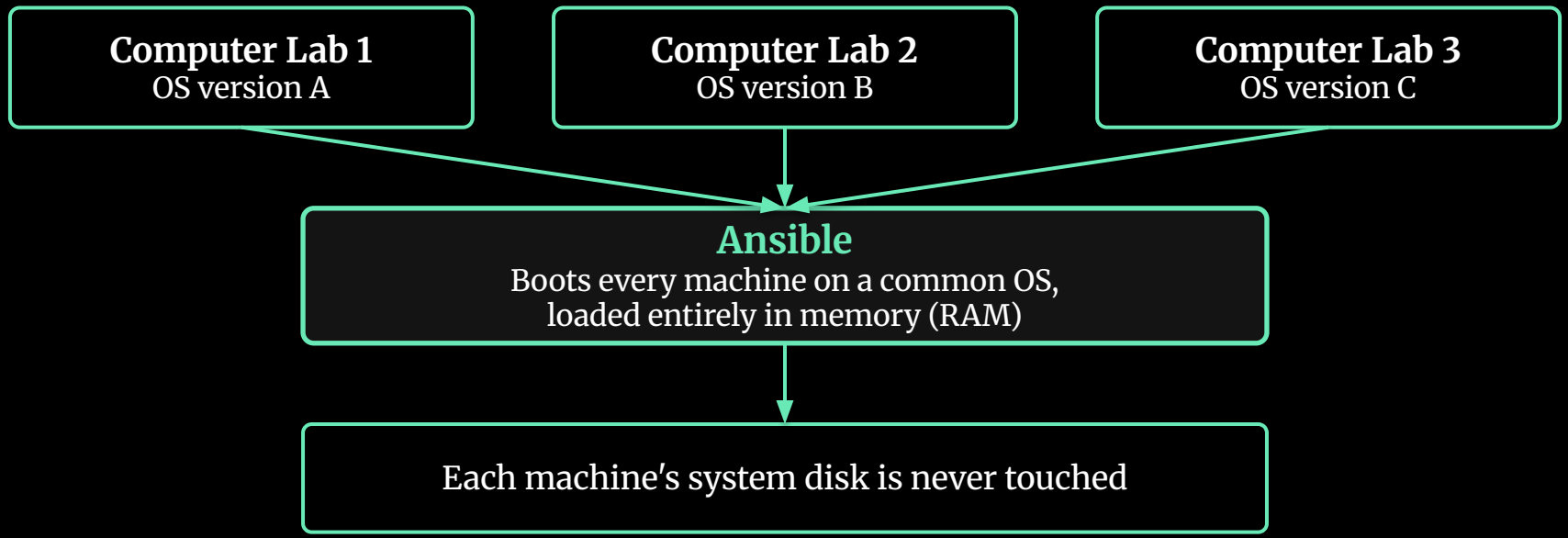
Computer Lab 2
OS version B

Computer Lab 3
OS version C

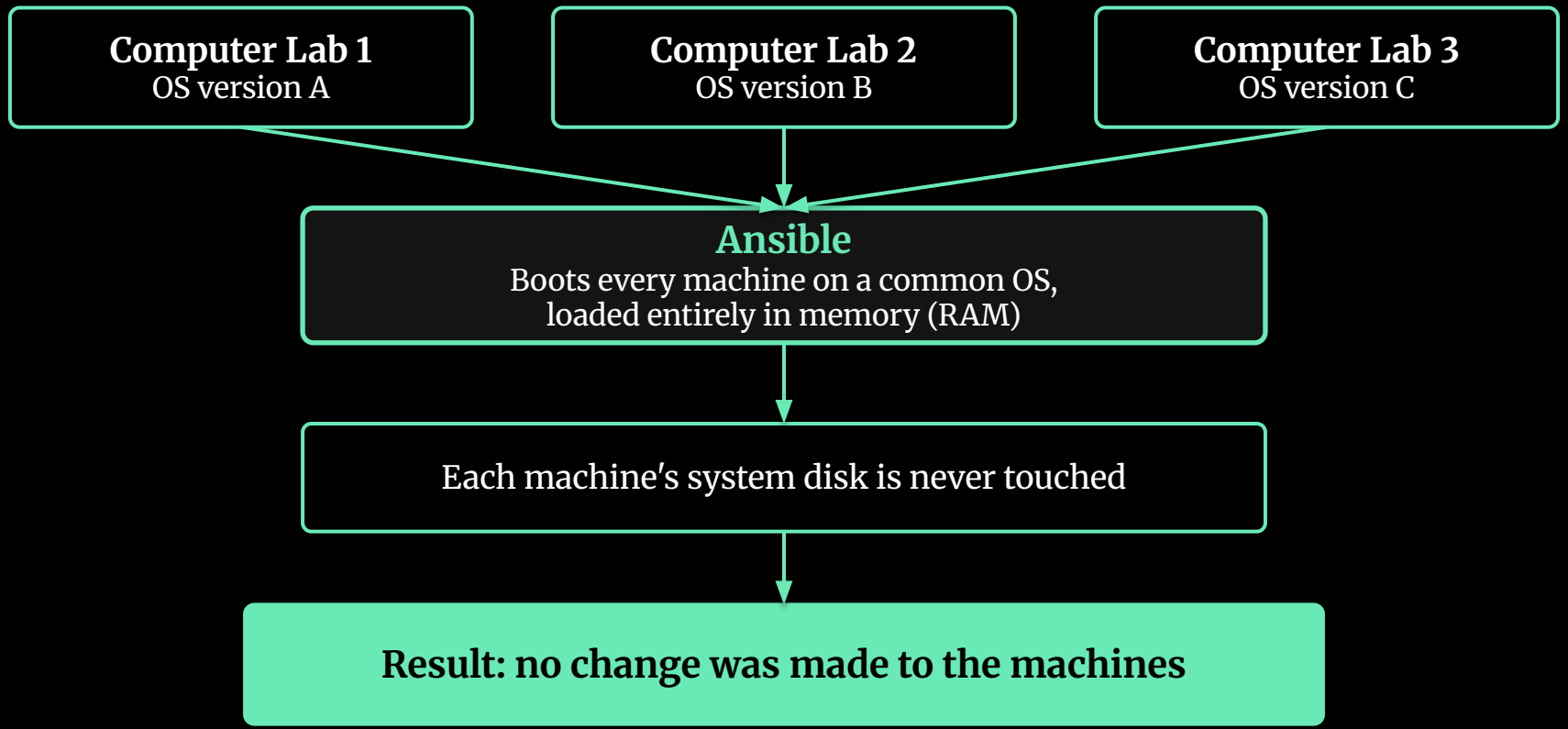
FASTER3 → Ansible Use case



FASTER3 → Ansible Use case



FASTER3 → Ansible Use case

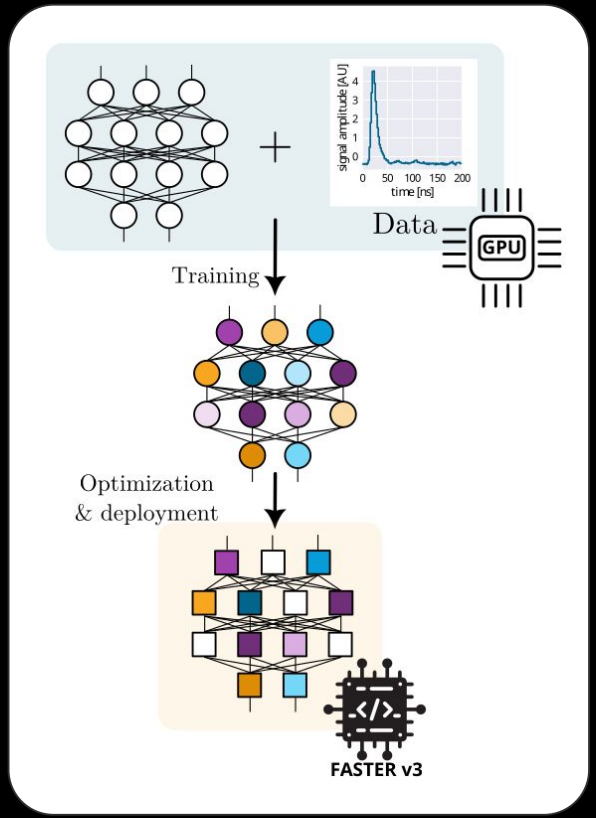


FASTER3 → Embedded AI models

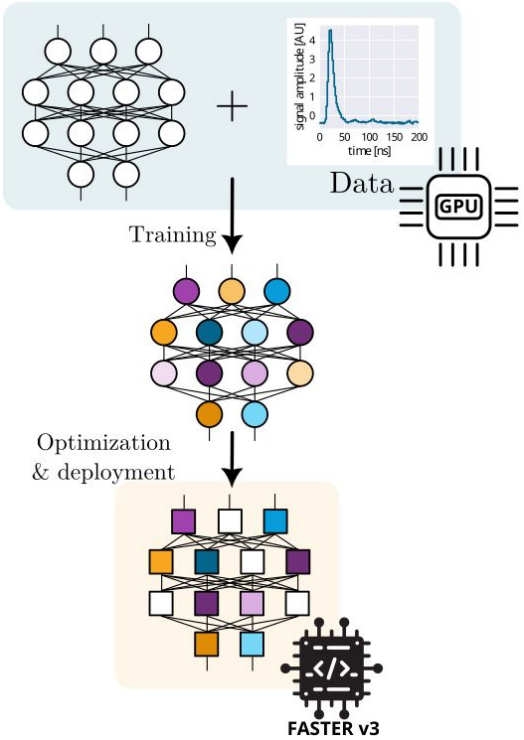
Thomas Carreau [carreau@lpccaen.in2p3.fr] LPC Caen, CNRS/IN2P3

FASTER3 → Embedded AI models

Thomas Carreau [carreau@lpccaen.in2p3.fr] LPC Caen, CNRS/IN2P3



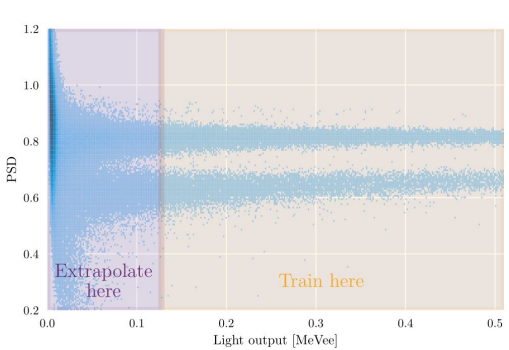
Thomas Carreau [carreau@lpccaen.in2p3.fr] LPC Caen, CNRS/IN2P3

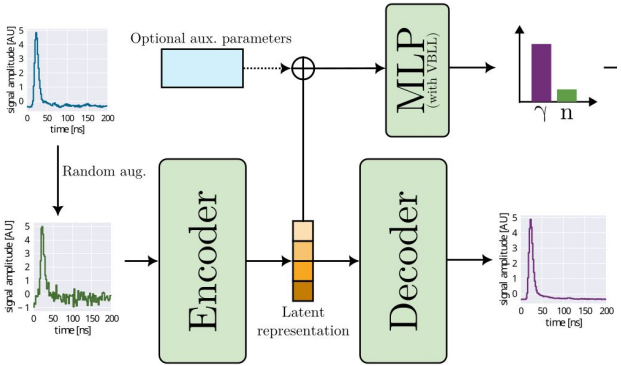


An example: neutron-γ discrimination

- Complete separation (FoM ≥ 3) above 100 keVee with PSD method. **Low SNR = unreliable PSD labels**.
- **Strategy:** We train our model to reconstruct and classify augmented high charge signals then extrapolate.

-> **Accurate** and **explainable** classification of low-charge signals.

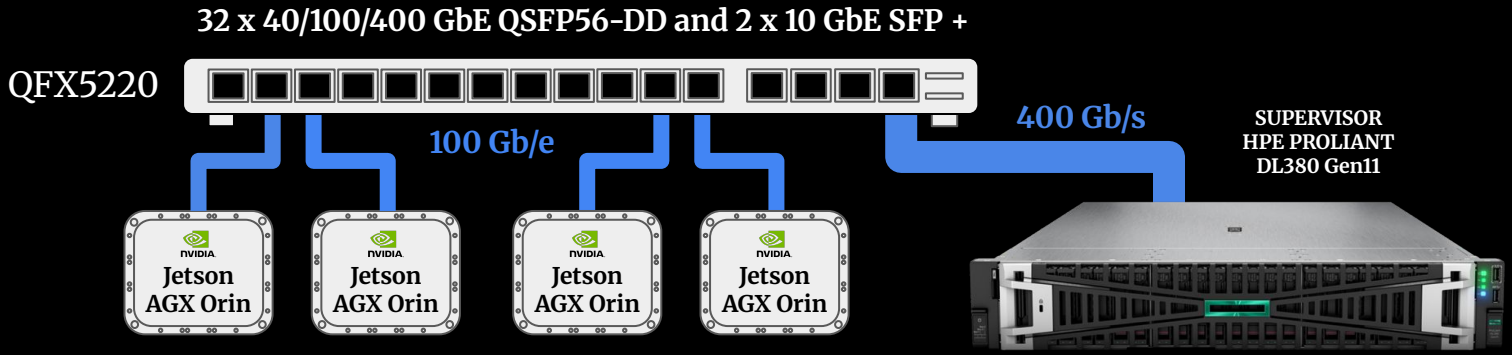




https://arxiv.org/pdf/2605.13627, accepted for publication in Phys. Rev. Research

Low-latency, high-throughput
packet processing
platform

Low-latency, high-throughput packet processing platform



Data Path → RDMA
Setup/Control Path → ZMQ

FASTER3 → ZMQ vs TCP/IP



ZMQ sockets are built on top of TCP/IP — it is not an alternative to TCP/IP, but a messaging layer on top of it

FASTER3 → ZMQ vs TCP/IP



ZMQ sockets are built on top of TCP/IP — it is not an alternative to TCP/IP, but a messaging layer on top of it

Raw TCP/IP sockets

- Byte stream — no message boundaries
- Manual reconnect & framing logic
- One blocking connection per socket

ZMQ (built on TCP/IP)

- Atomic messages — framing built in
- Automatic reconnect & queuing
- Built-in patterns, multiplexed I/O

FASTER3 → ZMQ vs TCP/IP



ZMQ sockets are built on top of TCP/IP — it is not an alternative to TCP/IP, but a messaging layer on top of it

Raw TCP/IP sockets

- Byte stream — no message boundaries
- Manual reconnect & framing logic
- One blocking connection per socket

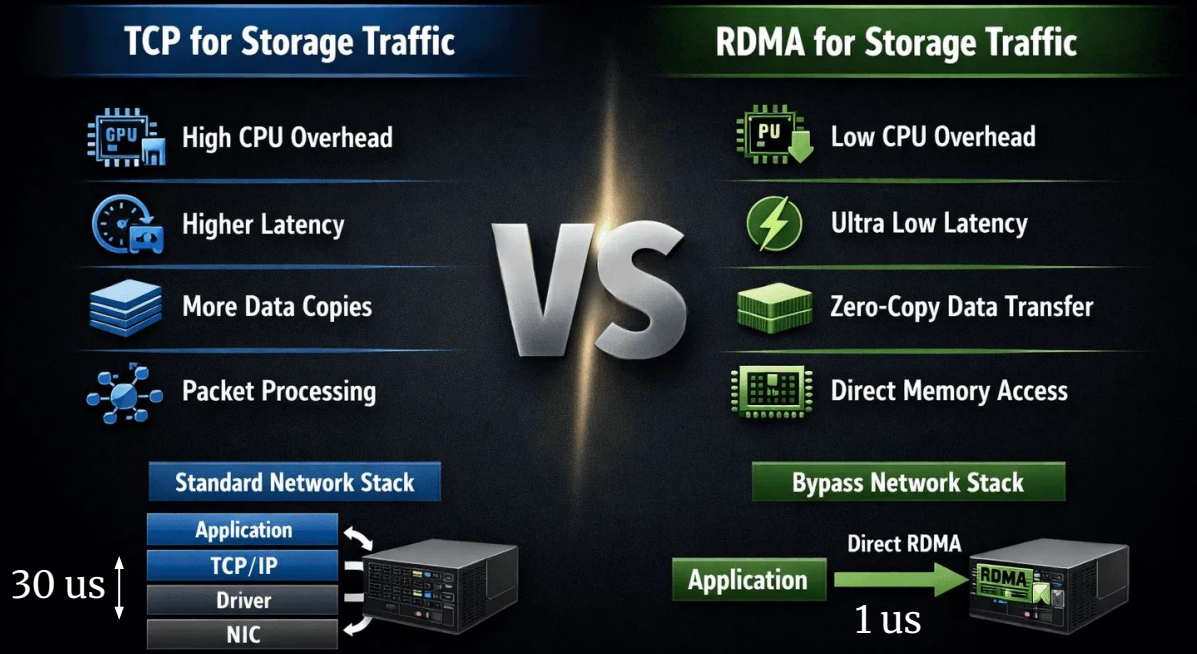
ZMQ (built on TCP/IP)

- Atomic messages — framing built in
- Automatic reconnect & queuing
- Built-in patterns, multiplexed I/O

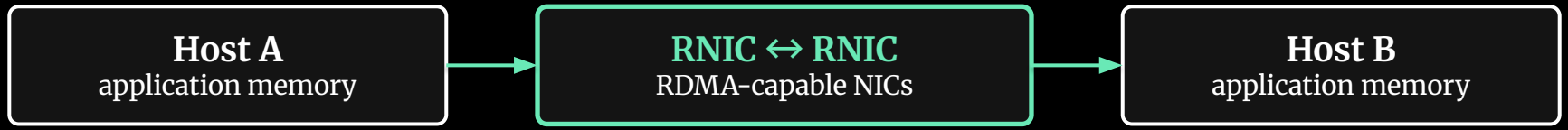
Overhead of ZMQ over raw TCP/IP

Small, sub-microsecond per-message cost (a few % for small messages, ~0 for large) — largely offset by ZMQ's internal batching, so throughput matches or exceeds raw sockets

FASTER3 → TCP/IP vs RDMA (Remote Direct Access Memory)



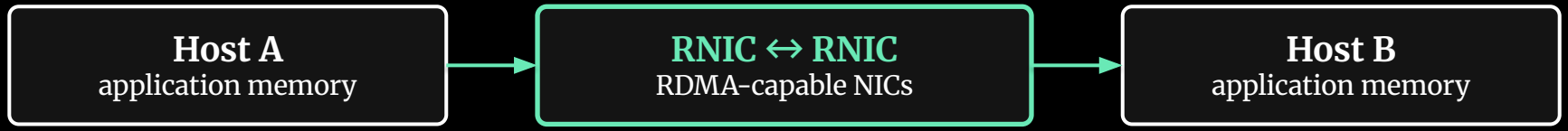
Basic principle



CPU and OS kernel are bypassed on both sides — zero-copy transfer, managed in hardware via Queue Pairs

FASTER3 → RDMA: RoCE & iWARP

Basic principle



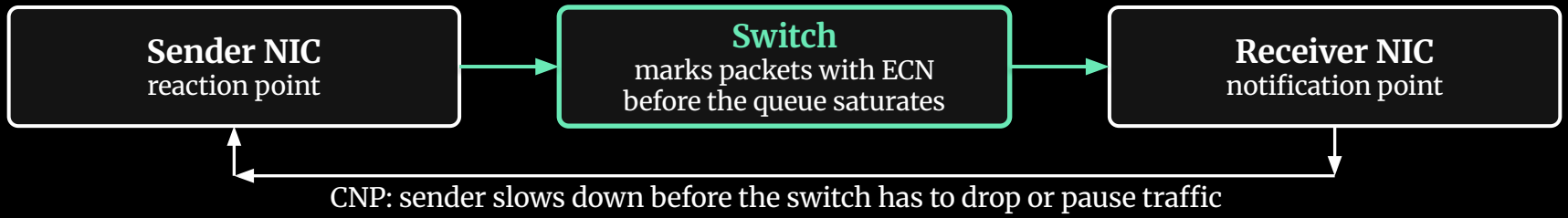
CPU and OS kernel are bypassed on both sides — zero-copy transfer, managed in hardware via Queue Pairs

RoCEv1, RoCEv2 & iWARP — FASTER3 uses RoCEv2 for Datapath

	RoCEv1	RoCEv2	iWARP
Network layer	Ethernet (L2)	**UDP / IP (L3)**	TCP / IP (L3)
Routable (L3)	No	**Yes**	Yes
Network requirement	Lossless Ethernet (PFC / DCB)	**Lossless Ethernet (PFC / ECN)**	Standard Ethernet (TCP handles loss)

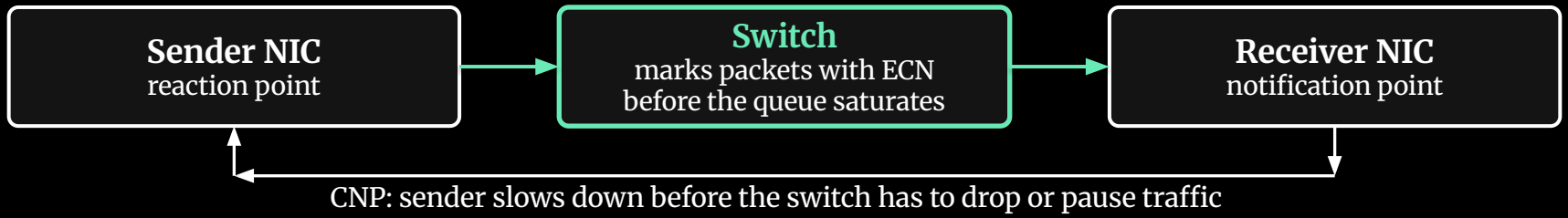
FASTER3 → Avoiding Saturation

RoCEv2 needs a lossless Ethernet fabric — a single dropped packet forces costly recovery



FASTER3 → Avoiding Saturation

RoCEv2 needs a lossless Ethernet fabric — a single dropped packet forces costly recovery



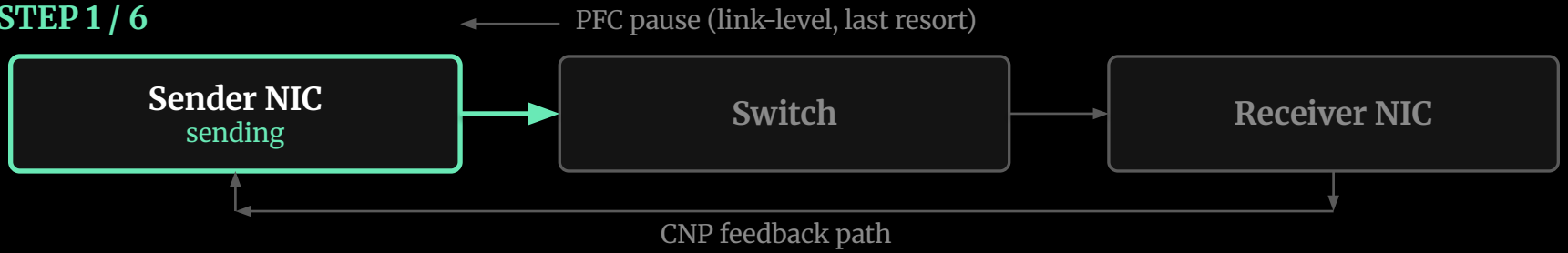
PFC — Priority Flow Control
Link-level backstop: pauses a queue before it overflows. Alone, it can cause pause storms and congestion spreading.

ECN — Explicit Congestion Notification
Switches mark packets as congestion builds, ahead of the PFC threshold — proactive, not reactive.

DCQCN — the closed loop that combines PFC + ECN
End-to-end rate control for RoCEv2: switches never need to drop packets — senders slow down before saturation occurs

FASTER3 → Congestion (1/6)

STEP 1 / 6

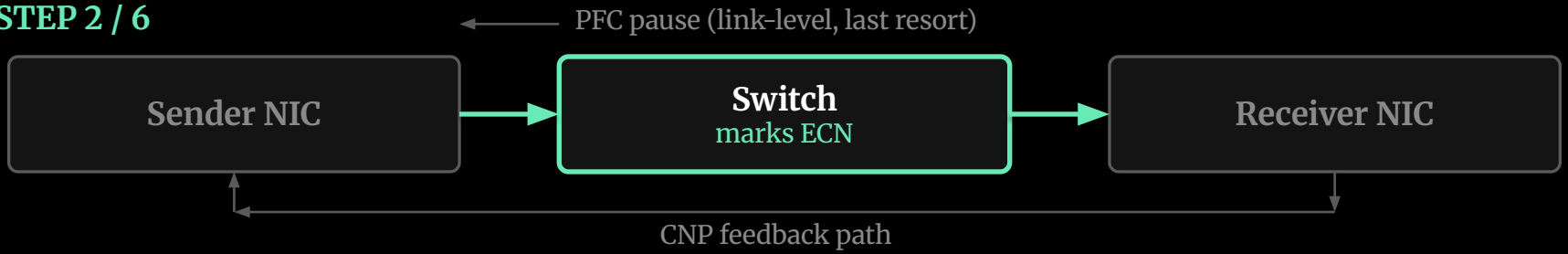


The sender NIC transmits data across the fabric as normal.

PFC	**ECN**	**CNP**	**DCQCN**
Priority Flow Control	Explicit Congestion Notification	Congestion Notification Packet	Data Center Quantized Congestion Notification

FASTER3 → Congestion (2/6)

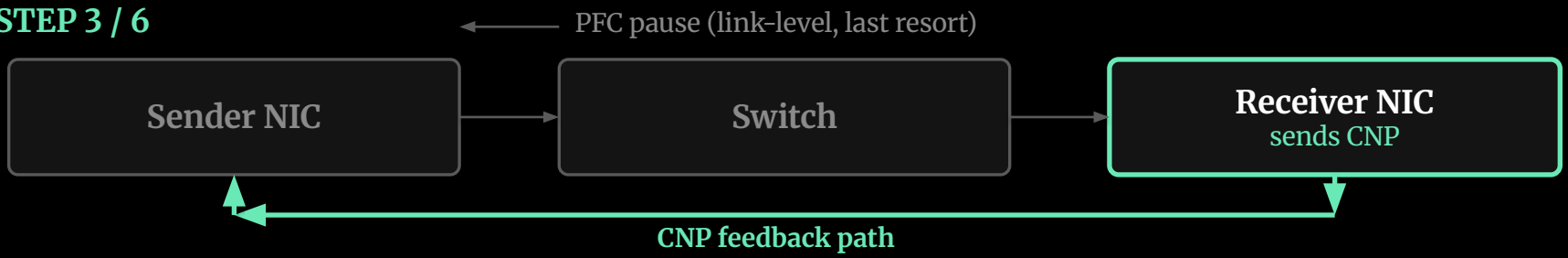
STEP 2 / 6



The switch queue starts to fill — it marks the packet with ECN before it saturates.

PFC Priority Flow Control	**ECN** Explicit Congestion Notification	**CNP** Congestion Notification Packet	**DCQCN** Data Center Quantized Congestion Notification

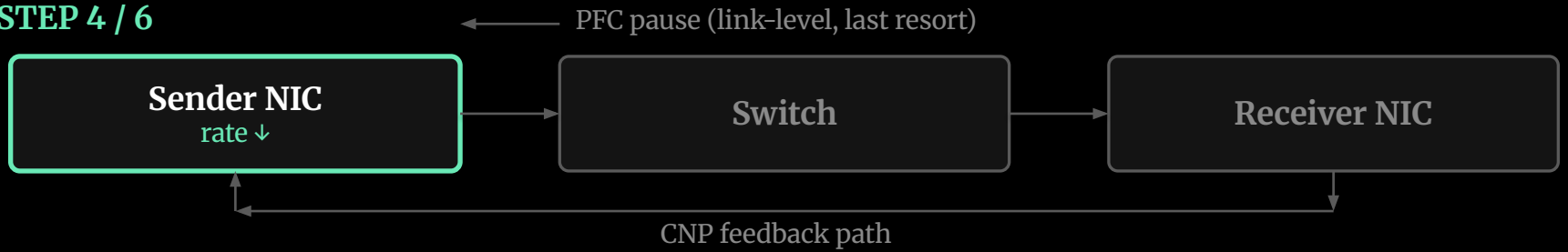
STEP 3 / 6



The receiver NIC sees the ECN mark and sends a CNP back to the sender.

PFC	ECN	CNP	DCQCN
Priority Flow Control	Explicit Congestion Notification	Congestion Notification Packet	Data Center Quantized Congestion Notification

STEP 4 / 6

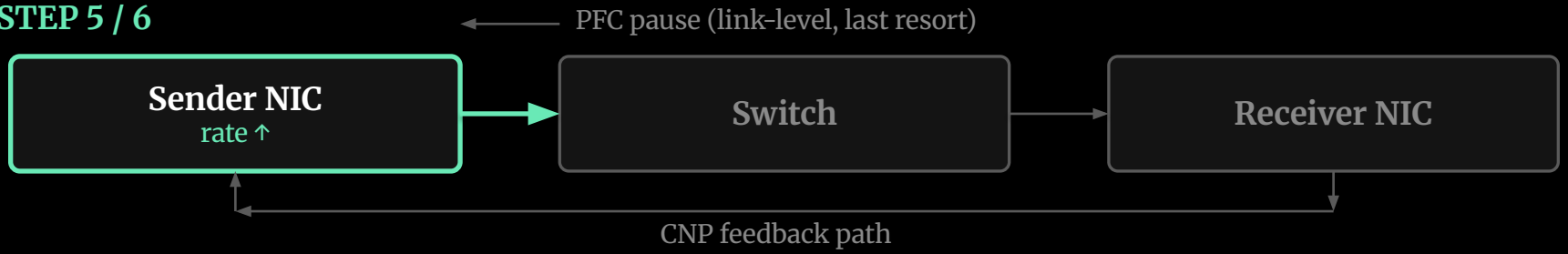


The sender NIC (DCQCN reaction point) receives the CNP and cuts its rate.

PFC Priority Flow Control	**ECN** Explicit Congestion Notification	**CNP** Congestion Notification Packet	**DCQCN** Data Center Quantized Congestion Notification

FASTER3 → Congestion (5/6)

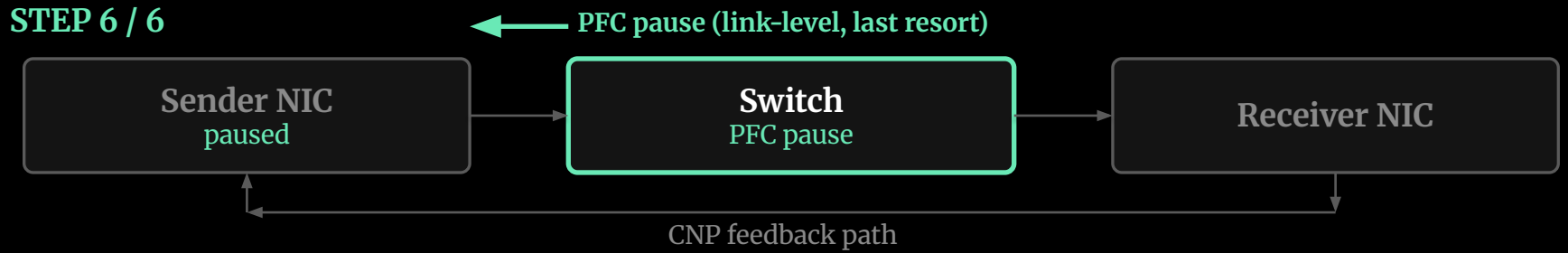
STEP 5 / 6



With no new CNPs, DCQCN gradually ramps the sending rate back up.

PFC	**ECN**	**CNP**	**DCQCN**
Priority Flow Control	Explicit Congestion Notification	Congestion Notification Packet	Data Center Quantized Congestion Notification

FASTER3 → Congestion (6/6)



If the queue keeps filling despite ECN/DCQCN, PFC pauses the link as a last-resort backstop.

PFC
Priority Flow Control

ECN
Explicit Congestion Notification

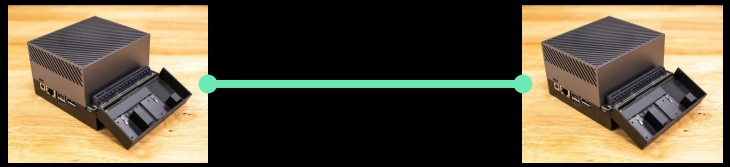
CNP
Congestion Notification Packet

DCQCN
Data Center Quantized Congestion Notification

FASTER3 → Jetson Orin Benchmarks

Goal
Measure the real throughput and CPU cost of each communication transport on SINAPSE hardware, to pick the right protocol for each data path

Test bed



2× Jetson AGX Orin 64GB
10G Ethernet + ConnectX-3 25 Gb/s RDMA

Transports tested

- inproc
- IPC
- TCP Local
- TCP Remote
- RoCE V2

Measured for each case:
throughput (Gb/s) and CPU load, sender / receiver

Results on the following slides

FASTER3 → Jetson AGX Orin 64GB



AI Performance
275 TOPS
INT8 sparse

GPU
2048-core Ampere
64 Tensor cores, 1.3 GHz

CPU
12-core
Cortex-A78AE
2.2 GHz, 3MB L2 + 6MB L3

Memory
64 GB LPDDR5
204.8 GB/s bandwidth

DL Accelerator
2x NVDLA v2.0
1.6 GHz

Vision Accelerator
PVA v2.0

Storage
64 GB eMMC 5.1

Power
15 – 60 W
MAXN performance mode

Jetson AGX Orin 64GB
2048-core Ampere GPU + 12-core Arm CPU in a single SoC

FASTER3 → ZMQ on One Machine



inproc

Jetson AGX Orin

Thread A → **Thread B**

shared memory — same process, no network stack

45.37 Gb/s

CPU (send / recv): 60.9% / 59.3%

FASTER3 → ZMQ on One Machine



inproc

Jetson AGX Orin

Thread A → **Thread B**

shared memory — same process, no network stack

45.37 Gb/s
CPU (send / recv): 60.9% / 59.3%

IPC

Jetson AGX Orin

Process A → **Process B**

Unix domain socket — local, no TCP/IP stack

14.07 Gb/s
CPU (send / recv): 95.6% / 79.2%

FASTER3 → ZMQ on One Machine



inproc

Jetson AGX Orin

Thread A → **Thread B**

shared memory — same process, no network stack

45.37 Gb/s
CPU (send / recv): 60.9% / 59.3%

IPC

Jetson AGX Orin

Process A → **Process B**

Unix domain socket — local, no TCP/IP stack

14.07 Gb/s
CPU (send / recv): 95.6% / 79.2%

TCP Local

Jetson AGX Orin

Process A → **Process B**

TCP/IP loopback (127.0.0.1) — full network stack, no NIC

11.68 Gb/s
CPU (send / recv): 89.7% / 63.5%

FASTER3 → ZMQ and RoCE V2 Across Machines

TCP Remote

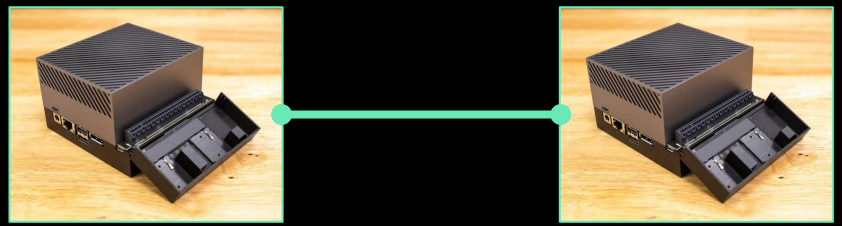
10G Ethernet

Jetson AGX Orin #1 ⟷ **Jetson** AGX Orin #2

TCP/IP over the 10G NIC

7.58 Gb/s

CPU (send / recv): 68.7% / 40%



RoCE V2

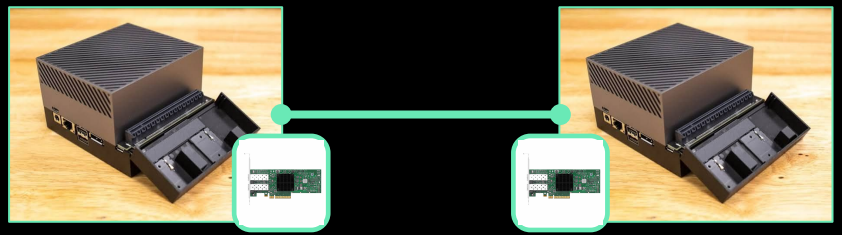
ConnectX-3, 25 Gb/s

Jetson AGX Orin #1 ⟷ **Jetson** AGX Orin #2

RDMA over Converged Ethernet v2 — receiver CPU stays at 0%

17.64 Gb/s

CPU (send / recv): 100.0% / 0.0%



FASTER3 → Project Motivations



FASTER3 → Project Motivations

More channels
per daughter-board
4 → 16 channels

New digitizers
125 Msps, 1,5Gsps@16bits
4Gsps@12bits

Cheaper



FASTER3 → Project Motivations

More channels
per daughter-board
4 → 24 channels

New digitizers
125 Msps, 1,5Gsps@16bits
4Gsps@12bits

Cheaper



Processing
FPGA, CPU, GPU

Embedded AI

Improved
performances
Synchronisation,
Accuracy, Throughput

FASTER3 → A synchronized multi-channel Oscilloscope module



FASTER2 → nptool v4: a tool for the community

PHILOSOPHY

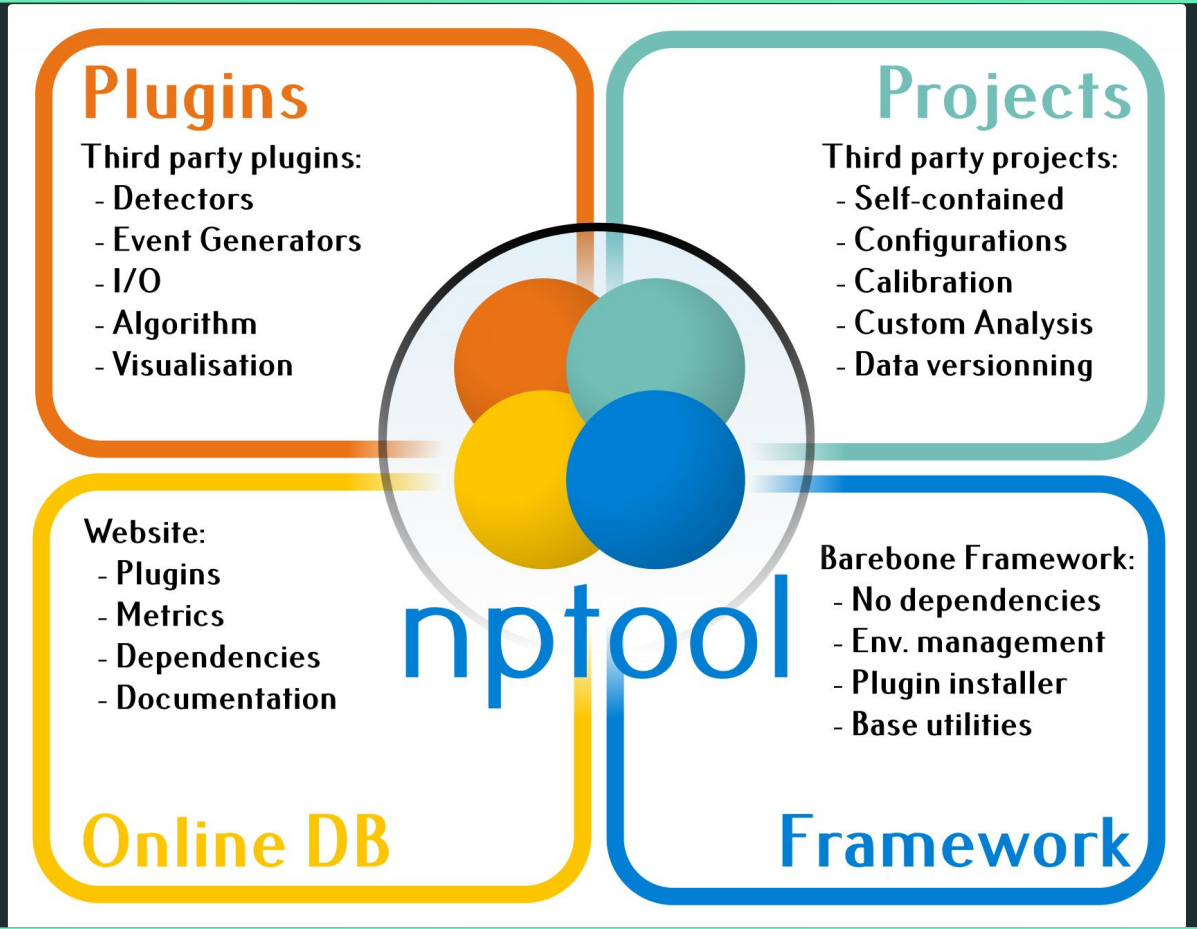
- Framework & Plugins
- Lightweight install
- Simplified InterOperability

FOR COLLABORATION

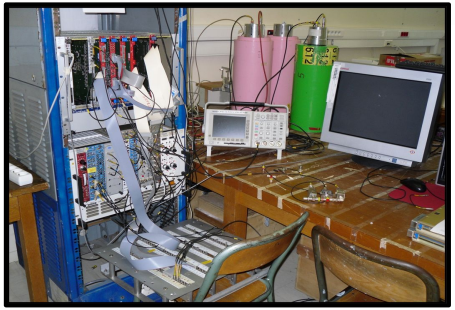
- Control their Plugin Repo
- EASY CI/CD
- Visibility w/ Citation Manag.

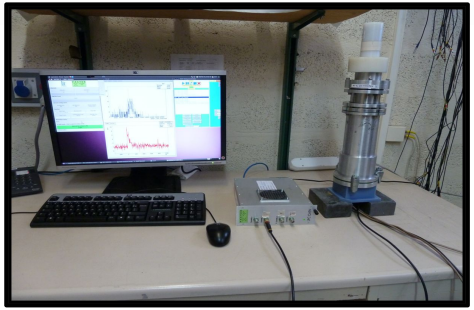
FOR USERS

- Plugin on demand
- Hassle free installation
- Custom Project
  - Custom Plugins
  - Custom Analysis
  - All in one place
- Open Science Ready
  - Metadata Production
  - Reproducible Analysis



FASTER2 → Historic











Setup/Control

faster_setup_gui
faster_gui
faster_daq_shell



Analysis (Online/Offline)

RHB
nptool
fasterac
pfasterac

2000 → 2008 FASTERv1 → 2012 FASTER2 → 2025

FASTERv2 has already been used with :

- Solid-state detectors (Silicon detectors, HPGe, SiLi and CdTe detectors),
- Gaseous detectors (Ionization chambers, GEM detectors, Proportional counters)
- Scintillators + Photomultiplier tube
  - Inorganic : NaI, CsI, YAG, LaBr, Li doped glass scintillator....,
  - Organic : Plastic scintillators, Liquid scintillators for pulse shape discrimination.
- RF cyclotron demodulation,
- SiPM detectors,
- Micro Channel plates with delay lines or resistive anodes for localization.