

LLRF 2026 Topical Workshop · Advanced Controls

When the Bare-Metal Returns

A Centralized Data Plane for Elettra 2.0 Control and Simulation

Giulio Gaio

Alessio Bogani, Gabriele Brajnik, Igor Trovarelli, Marco Cautero, Raffaele De Monte

Elettra Sincrotrone Trieste

1

Why bare metal

data efficiency and jitter

2

Real time on Linux

isolated cores, Ethernet, DPDK

3

Validated in production

the Elettra fast orbit feedback

4

Elettra 2.0

the RT cluster and its test facility

5

Hunting jitter

packets as bunches

6

CRUDA and the digital twin

the whole machine in one memory

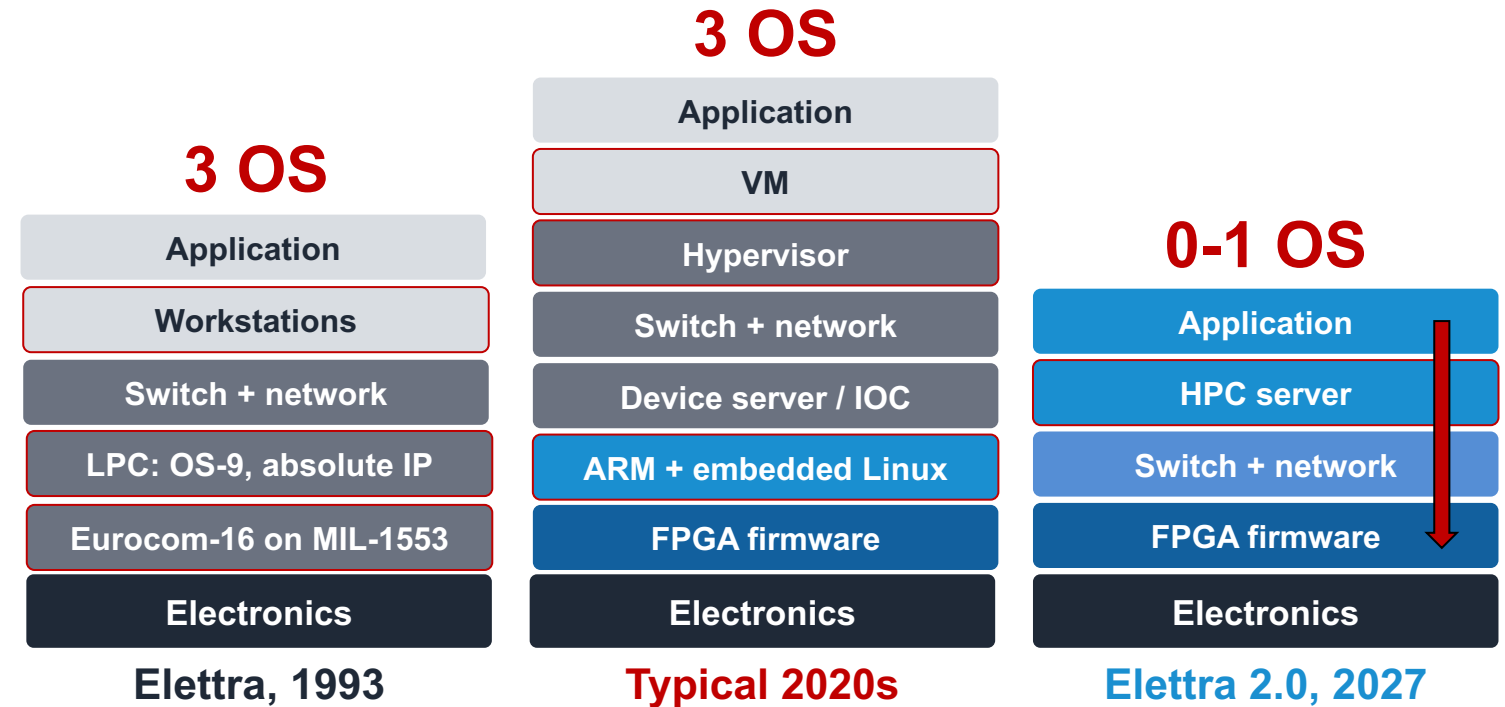
7

What comes next

High availability not from day zero

Why bare metal: more data to digest, fewer layers to cross

- Data-driven programs now sit beside model-driven ones: much more data to digest.
- Every layer the data cross, an **O**perating **S**ystem above all, makes processing efficiency collapse.
- **Bandwidth you can buy. Jitter you must tune.**



Six time domains, each set by its jitter. Faster data stay near the hardware: automation must go down.



Each system repeats at the rate its own jitter allows: the domains are physics.

Real-Time and Linux: CPU isolation, Non-Uniform Memory Access (NUMA) architecture

RTAI, Xenomai

2003–2019: co-kernels, an island per loop



PREEMPT_RT

preemptible locks, threaded IRQs; mainline in 6.12

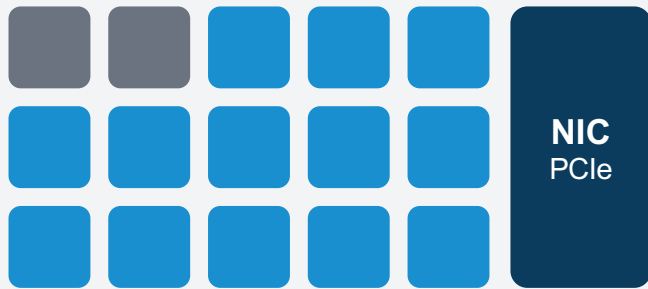


Today: isolated cores

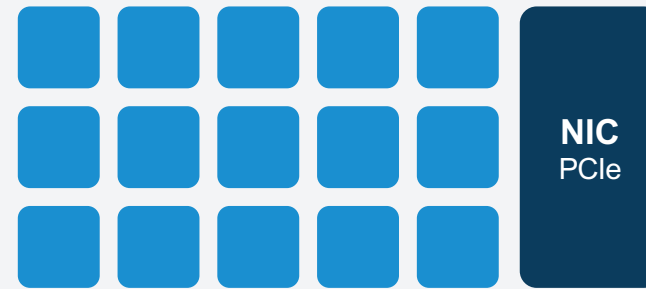
the RT task owns its core: no tick, no IRQs

Dual-socket server: partition it, then keep everything local

Socket 0 · NUMA node 0



Socket 1 · NUMA node 1



Isolation knobs

isolcpus no load balancing
nohz_full no scheduler tick
rcu_nocbs RCU callbacks off
irqaffinity IRQs off RT cores
HugePages fewer TLB misses
BIOS deep C-states off

■ Housekeeping cores: Linux, IRQs, RCU callbacks, services

■ Isolated cores: one RT thread each, no tick, no IRQs

Today: rebuild a mono-core inside a multi-core processor

Ethernet: one bus, convenient, eternal, pervasive

Convenient

Commodity NICs, switches, optics.
Multi-vendor.
Standard tools: tcpdump, Wireshark,
port mirroring.

Pervasive

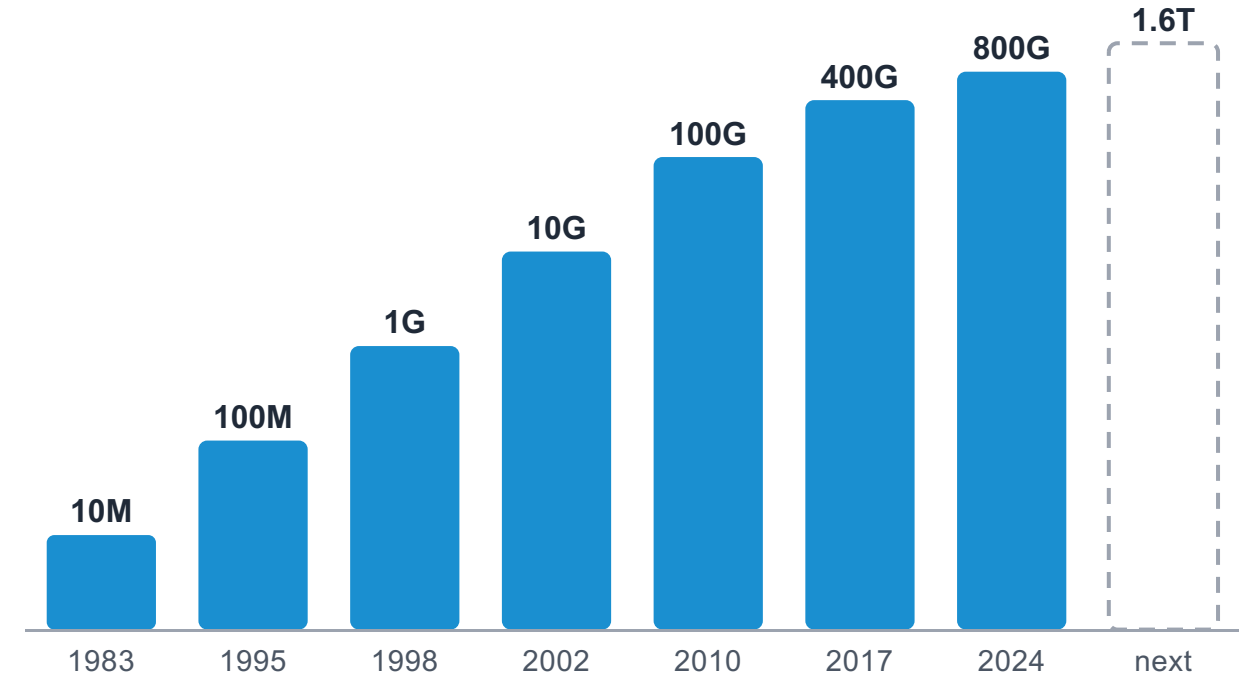
Every FPGA, SoC and server has a
MAC.
10 Gb/s is the floor, 100 Gb/s
common, 400 Gb/s commodity.

Eternal

IEEE 802.3 since 1983.

Every generation uses the same
frame format.

Ethernet speed, IEEE 802.3 *log scale*



Same frame, 1983 → 2024: 80,000 × faster

Buses come and go. Ethernet stays.

MIL-1553B (1993), the reflective memory ring (2007), VME backplanes: retired. Elettra 2.0: 160 Mpps of UDP into one server.

DPDK: bypassing the network kernel layer

Principles

Kernel bypass: user-space drivers

Poll mode: no interrupts

Cores isolated, pinned to NIC queues

NUMA aware

Vectorized / SIMD

Cache aligned

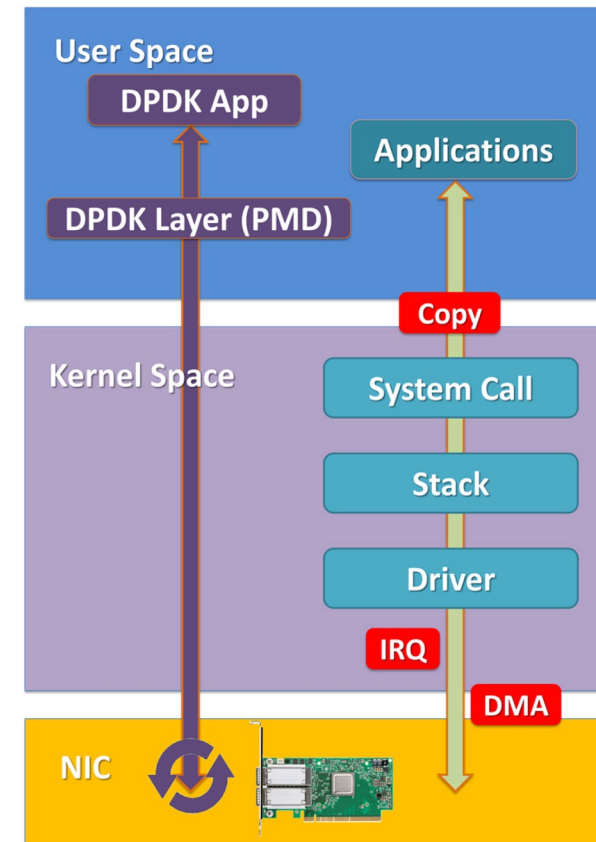
Plain C

DPDK *the first brick*

Industrial grade

Intel 2013 → Linux Foundation · 5G user plane (UPF) · AWS, Azure, Google Cloud

Kernel path vs DPDK path



Gold Members



ZTE

Silver Members



Associate Members

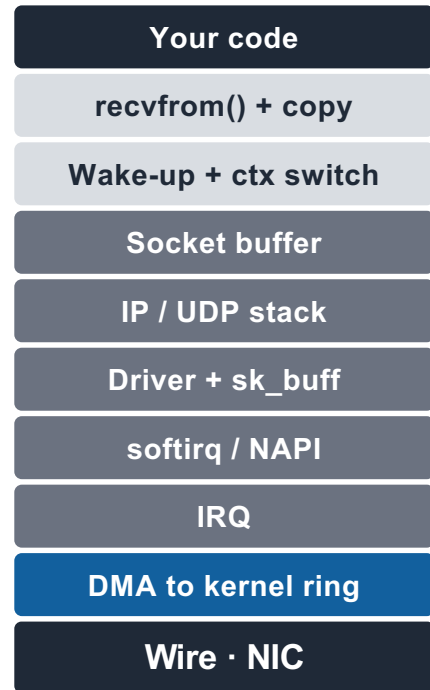


No interrupts, no system calls, no copies: the NIC talks to CPU core.

A vendor-neutral data plane maintained by others: no more drivers patched at every kernel upgrade.

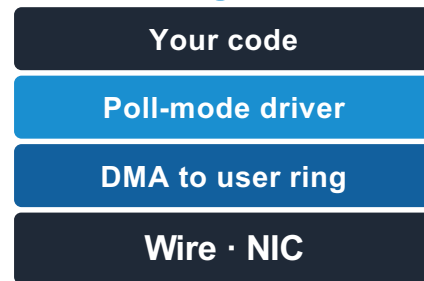
Socket vs DPDK: counting layers again in the Round-Trip Time test

8 layers

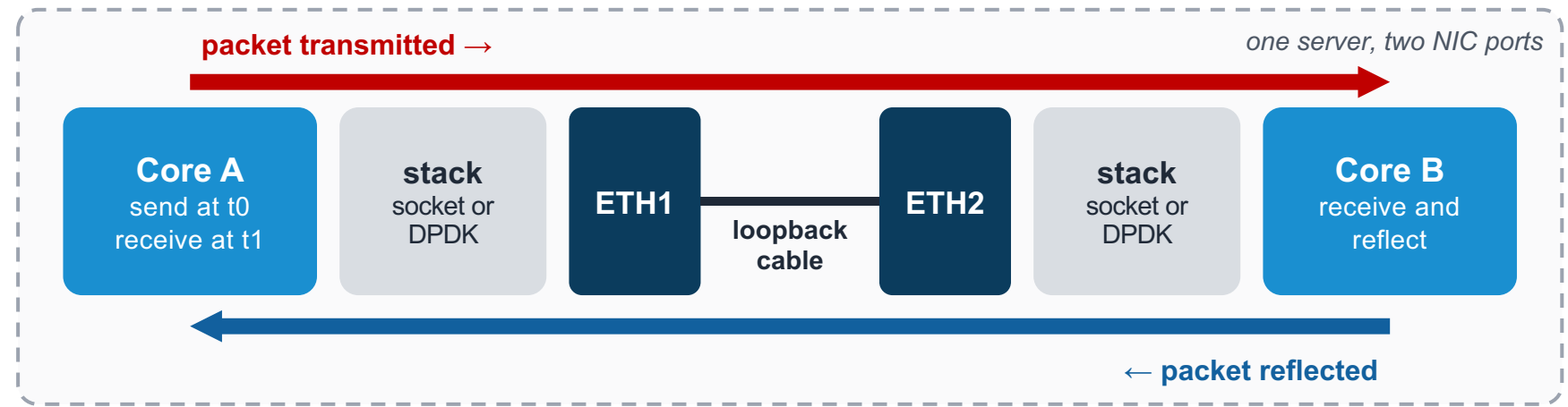


Linux UDP socket

2 layers



DPDK



$RTT = t1 - t0$ one-way latency $\approx RTT / 2$ a control loop in miniature

RTT under load	mean	max
Raw socket	16.9 μs	6768 μs
Raw socket, isolated core	23.9 μs	191 μs
DPDK	8.9 μs	60.6 μs
DPDK, isolated core	9.0 μs	13.7 μs

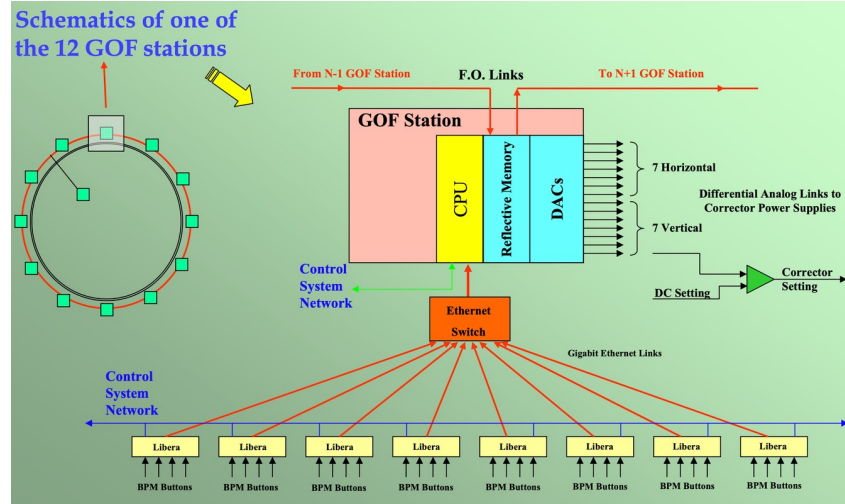
Loopback between two NIC ports, TX and RX pinned to dedicated cores, stress-ng on CPU, cache and RAM. Xeon Gold 6346, Intel X710, DPDK 22.11 (IBIC 2025).

Fewer layers, less jitter

And the raw socket is the best case for EPICS / Tango: every conventional control system sits above it.

DPDK in production (1): Fast Orbit Feedback, 2007 - 2024

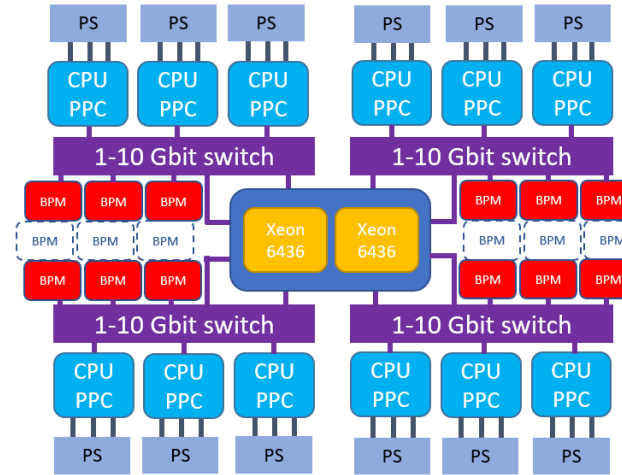
2007 distributed



1 Mpps
13 VME CPUs
+ reflective memory ring

- 12 VME stations + 1 master (MVME6100), 96 BPMs at **10 kHz**
- SVD + PID + 6 harmonic suppressors in every station
- Reflective memory ring (3.125 Gb/s) shares the orbit

2024 centralized



10 Mpps
6 BPM electronics
+ LLRF prototype
acquired turn by turn
1 server, everything in RAM

- 1 dual-Xeon server (16+16 cores), DPDK
- 88 Libera BPMs at 10 kHz; 6 eBPMs + LLRF prototype at **1.15 MHz**
- SVD + PID + 13 harmonic suppressors, all in RAM
- No reflective memory; VME boards only drive the DACs

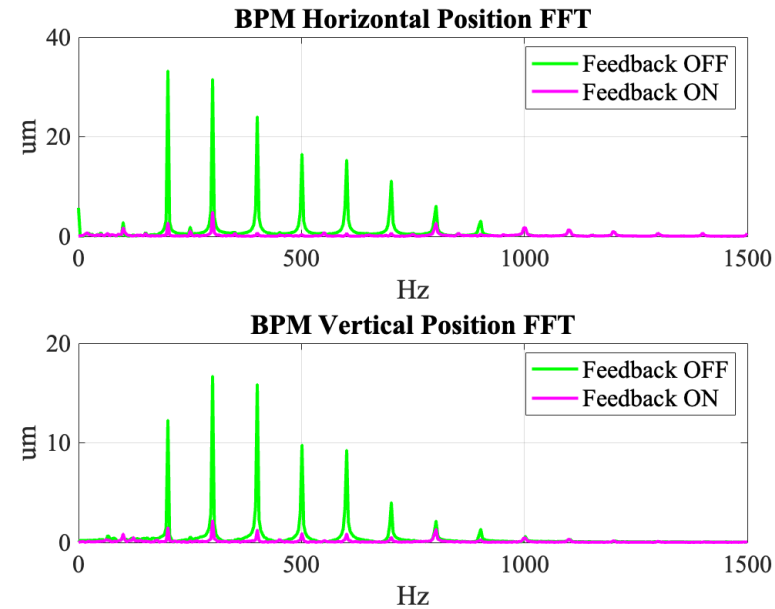
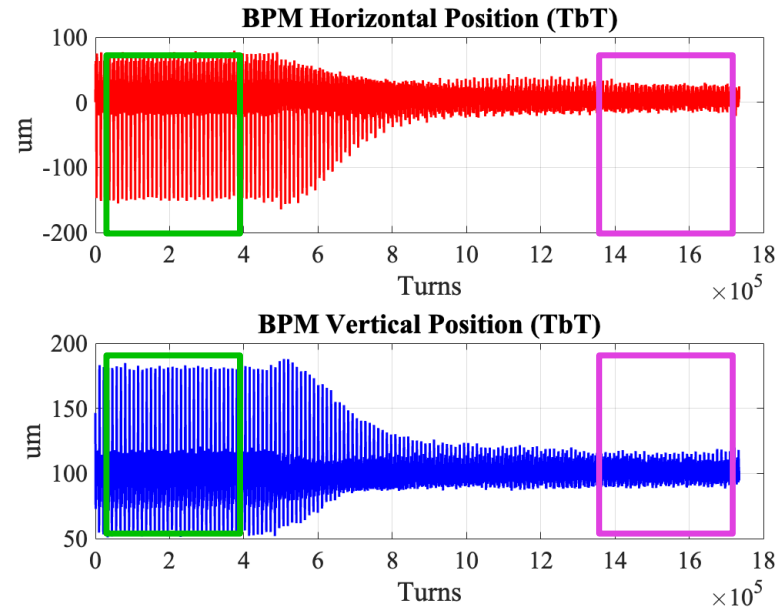
Validated in production: 12 stations + a memory ring → one server.

10 × the packets, faster streams (1.15 MHz) on the same machine, and fewer boxes to keep alive.

DPDK in production (2): orbit correction up to 900 Hz

Final months of Elettra operation: the power-supply filters had nearly failed.

The GOF corrected distortions up to 900 Hz, with harmonic suppressors at 50, 100, 150, 200, 250, 300, 350, 400, 500, 600, 700, 800, 900 Hz.



Green window: feedback OFF Magenta window: feedback ON

8 μ s

physics per 10 kHz cycle, one core:
InvORM 82×96 , PID, harmonic
suppressors

100 GB

circular buffers in RAM: 10 kHz and
1.15 MHz data

10 s

empirical orbit response matrix

2 kHz

tune refresh from turn-by-turn data:
moving 8k FFT, overlapping
windows

Next: Elettra 2.0, about 528 power supplies at 100 kHz, a 10 μ s cycle.

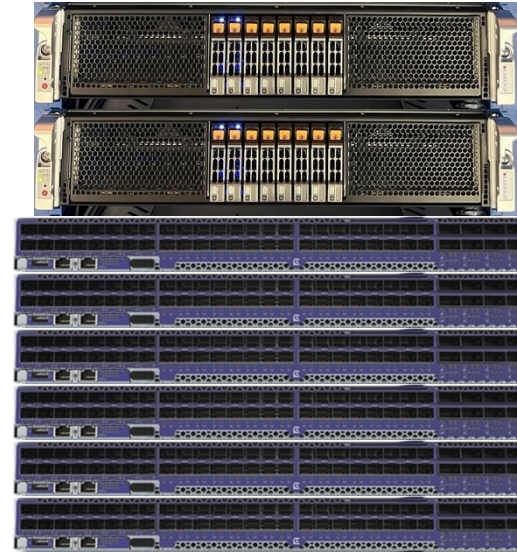
Elettra 2.0: the conventional and the RT clusters

Conventional cluster slow services · 10 GbE



- Network services, logging, configuration
- Slow monitoring: PLC, vacuum systems, insertion devices, ...
- 10-Gbit links to the edge switches

RT cluster fast control · 100 GbE · in RAM



- Fast control, orbit feedback, digital twin
- Turn-by-turn storage on events, pre/post mortem, online logic and analysis
- Every device that directly affects machine performance: read/set ≥ 1 kHz
- **All data in RAM:** an in-memory control system
- 100 Gbit to the RT servers, 10 Gbit to sensors / actuators: **FPGA direct connection, SoC bypass**

Two control clusters: no VME, no cPCI, no MicroTCA.

The conventional cluster runs the slow services; the RT cluster keeps every fast signal of the machine in RAM, straight from the FPGAs.

RT control system: computing power and network load

RT server Hyper A+ Server AS-2126HS-TN

- 2 × AMD EPYC 9755: 128 + 128 cores (256 + 256 HT)
- 1.5 TB DDR5-6400 RAM
- 1 dual Mellanox ConnectX-5 100 Gbit: to the control system
- 3 Intel E810 dual 100 Gbit: to sensors / actuators

Switch Extreme Networks 7520

- 4 Tb/s max capacity
- 48 × 10-Gbit ports, 8 × 100-Gbit ports

2 servers

1024 HT cores,
12 × 100-Gbit ports

6 switches

288 × 10-Gbit,
48 × 100-Gbit ports

Source	Num.	Rate	Mpps
BPM electronics	84	1.157 MHz	97
BLM electronics	12	1.157 MHz	14
LLRF	4	1.157 MHz	5
Booster electronics	10	2.41 MHz	24
PS chains (≤ 14 PS)	96	200 kHz	19
Photon diagnostics	–	to come	–
Total in			≈ 160

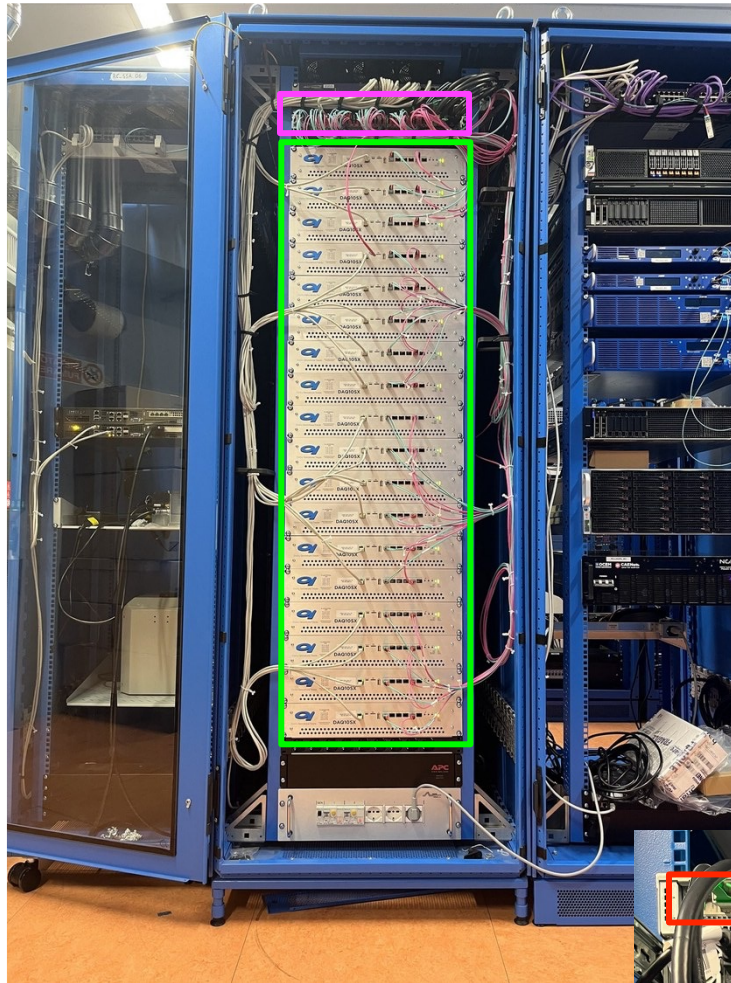
≈ 160 Mpps
in, every source at its
own rate

10–20 Mpps
out: setpoints to the 96
PS chains

1.157 MHz
the turn: BPM, BLM and
LLRF

Total cost = 270 k€: the fast data of the whole machines.

RT-Cluster Test Facility: testing BPMs, PS and RT-servers

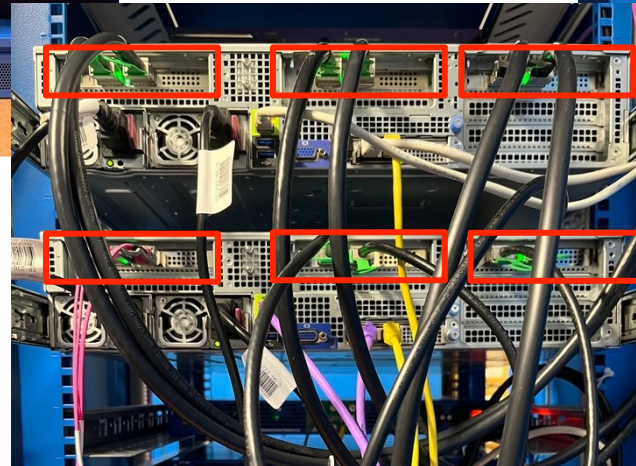
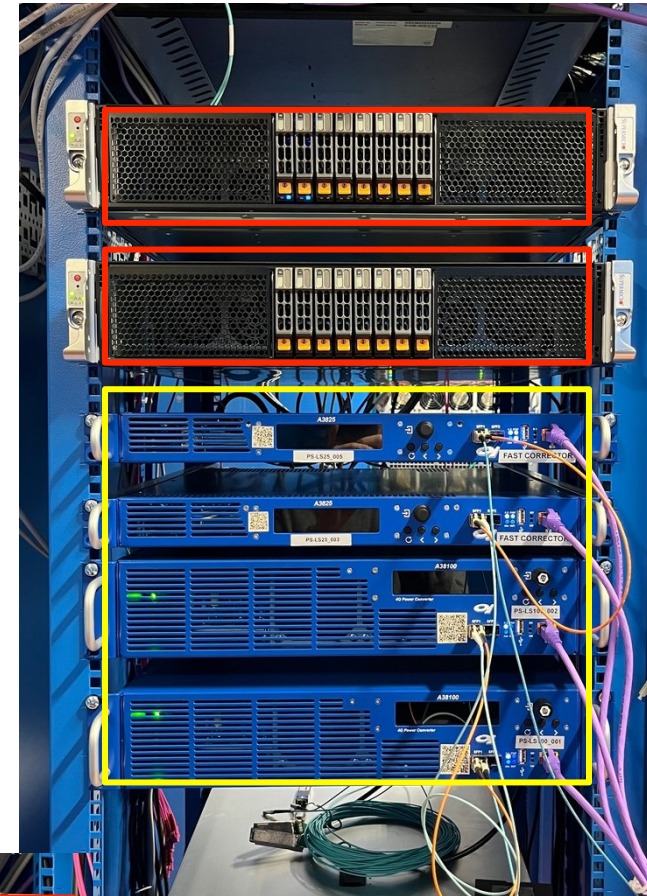


18 new eBPMs
transmitting UDP at
1.15 MHz

A switch mirrors 18
BPM streams onto
six 100-Gbit ports
Up to 108 Mpps

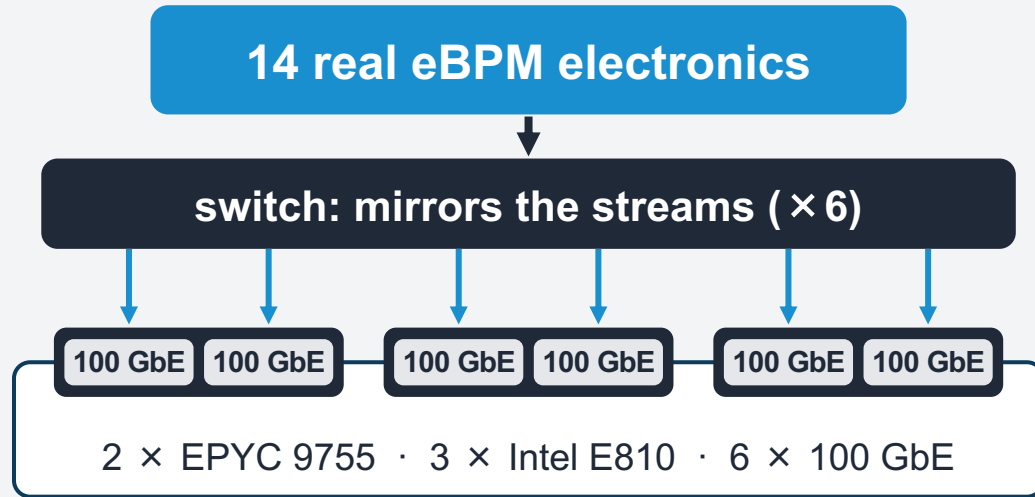
RT-servers
600 Gbit IN/OUT

Daisy chain of 4
power supplies
200 kHz rep-rate



The setup: two servers, different loads

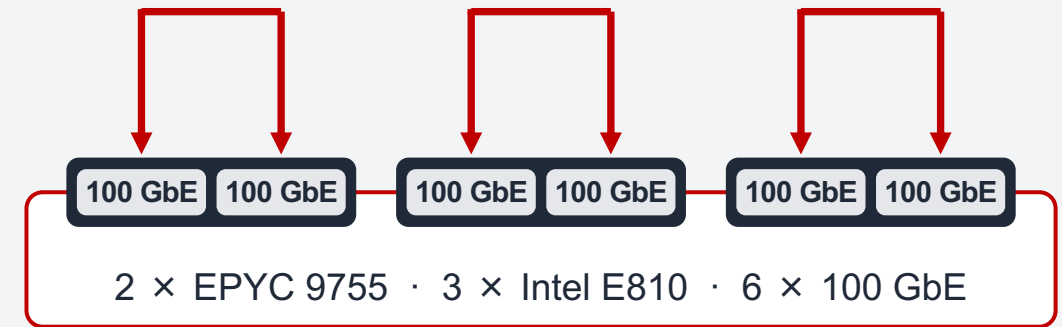
fast-01 the storage ring real electronics



97 Mpps in 84 streams, one packet per turn:
almost the real load of the machine

fast-02 six ports in loopback

the two ports of each card, cabled together
what one transmits, the other receives



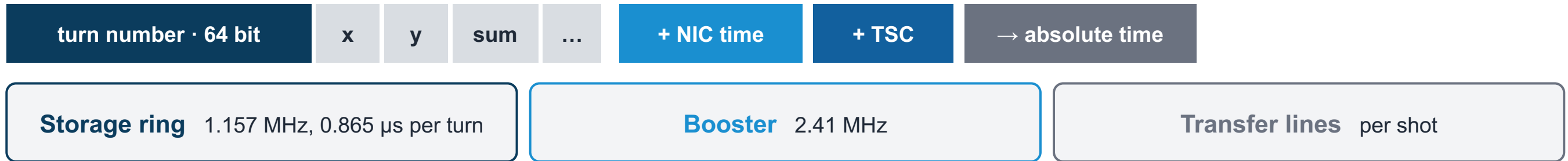
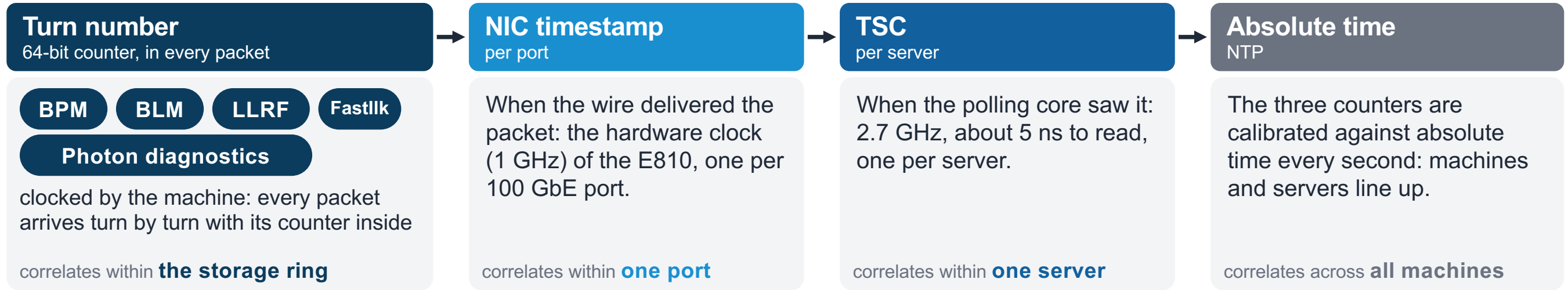
**195 Mpps in
+ 195 Mpps out**

32 Mpps per port, each way:
synthetic, more than the
whole machine

fast-01 listens to the real BPMs; fast-02 talks to itself, at x2 the design load.

Twin servers, same hardware, same software: develop on fast-02, long runs on fast-01.

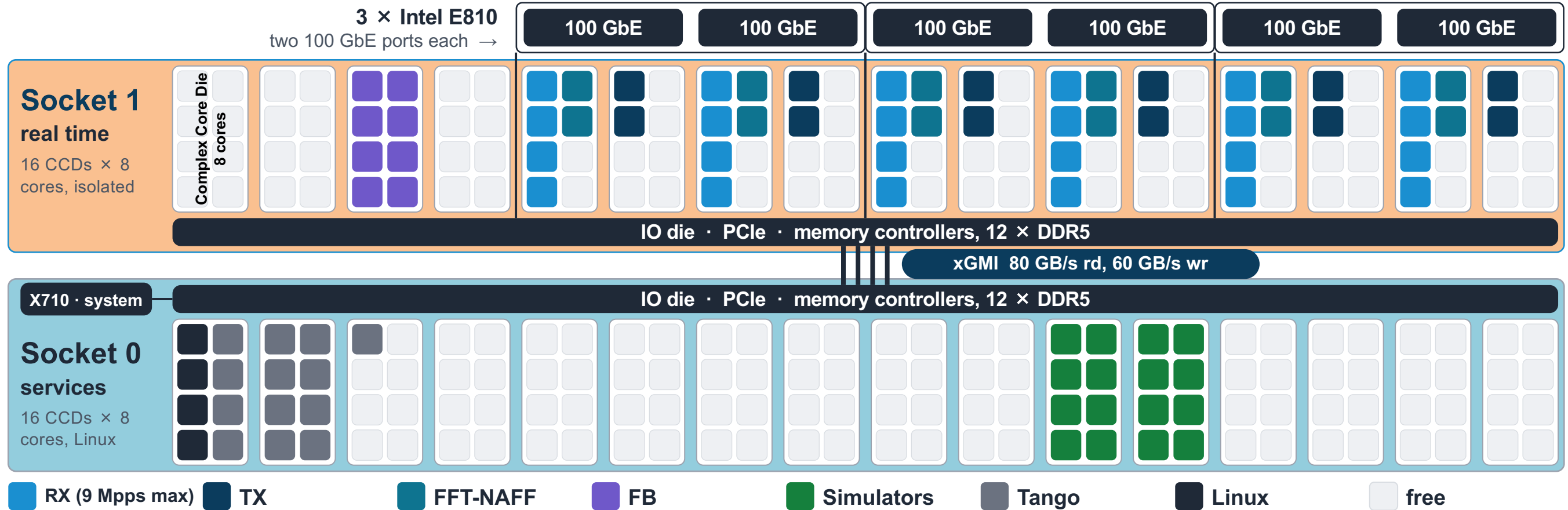
Timing: every packet carries multiple timestamps



Data are searched on every time base, to correlate sources at different rates.

Turn number, NIC clock, TSC and absolute time travel with every record, each with its own reach: the turn number only inside the storage ring, the TSC inside one server, absolute time across all machines (control room workstations).

Inside the server: how the load is distributed



Each 100 GbE port has its own two chiplets: one for RX, one for TX.

Socket 1 is the data plane, isolated: RX, TX and FFT-NAFF for every port, and the feedback (FB).

RX cores: TbT (1.157 MHz) → 100 kHz and 10 Hz, decimated on the fly with AVX-512.

Socket 0 holds the services: Linux, the Tango servers, the model. Between them, xGMI.

Server typical load: ~83 of 512 cores busy

```
%Cpu(s): 16.1 us, 0.0 sy, 0.0 ni, 83.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1547554.+total, 1006218.+free, 546423.4 used, 2019.2 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1001131.+avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
351954	root	rt	0	386.9g	25372	16244	R	1117	0.0	4d+19h	dpdk_master
351743	root	rt	0	386.9g	25380	16244	R	1117	0.0	4d+19h	dpdk_master
351783	root	rt	0	386.9g	25380	16244	R	1117	0.0	4d+19h	dpdk_master
351822	root	rt	0	386.9g	25376	16244	R	1117	0.0	4d+19h	dpdk_master
351876	root	rt	0	386.9g	25396	16244	R	1117	0.0	4d+19h	dpdk_master
351915	root	rt	0	386.9g	25380	16244	R	1117	0.0	4d+19h	dpdk_master
353323	root	20	0	66.1g	9.5g	24588	S	318.0	0.6	33,15	catds-ref-srv
351989	root	rt	0	134.0g	2996	2604	R	100.0	0.0	10,30	dpdk_ebpm
352712	root	rt	0	2734536	8348	3048	R	100.0	0.0	10,29	naff-ccd
352788	root	rt	0	2734536	8352	3048	R	100.0	0.0	10,29	naff-ccd
352678	root	rt	0	2734536	8348	3048	R	100.0	0.0	10,29	naff-ccd
352731	root	rt	0	2734536	8348	3048	R	100.0	0.0	10,29	naff-ccd
352750	root	rt	0	2734536	8348	3048	R	100.0	0.0	10,29	naff-ccd
352769	root	rt	0	2734536	8348	3048	R	100.0	0.0	10,29	naff-ccd
352822	root	rt	0	2734536	8352	3048	R	100.0	0.0	10,29	naff-ccd
352841	root	rt	0	2734536	8352	3048	R	100.0	0.0	10,29	naff-ccd
352860	root	rt	0	2734536	8352	3048	R	100.0	0.0	10,29	naff-ccd
352879	root	rt	0	2734536	8352	3048	R	100.0	0.0	10,29	naff-ccd
352898	root	rt	0	2734536	8352	3048	R	100.0	0.0	10,29	naff-ccd
352917	root	rt	0	2734536	8352	3048	R	100.0	0.0	10,29	naff-ccd
353070	root	20	0	128.0g	30980	16224	S	1.7	0.0	9:52.96	tcbuf-srv
433508	root	10	-10	13892	11504	3848	R	0.7	0.0	0:00.15	top
352958	root	20	0	8083200	71516	17664	S	0.3	0.0	0:31.72	devx-srv
353670	root	20	0	128.5g	28936	16928	S	0.3	0.0	0:13.68	devx-srv
353693	root	20	0	128.5g	28936	16928	S	0.3	0.0	0:15.76	devx-srv
422683	root	20	0	0	0	0	I	0.3	0.0	0:00.22	kworker/u2082:15-xprtiod
1	root	20	0	22416	13484	9600	S	0.0	0.0	0:05.45	systemd

top, during the measurement · %CPU summed per class (100 % = one core)

I/O Communication · 6 ports

6702 %

Simulators **318 %**

TbT FFT

1200 %

Everything else

< 4 %

control system, Linux

total ≈ 8320 % ≈ 83 cores of 512 (16 %)

+ 10 MB/s networking

Packets as bunches: measuring the TbT inter-packet time to detect jitter

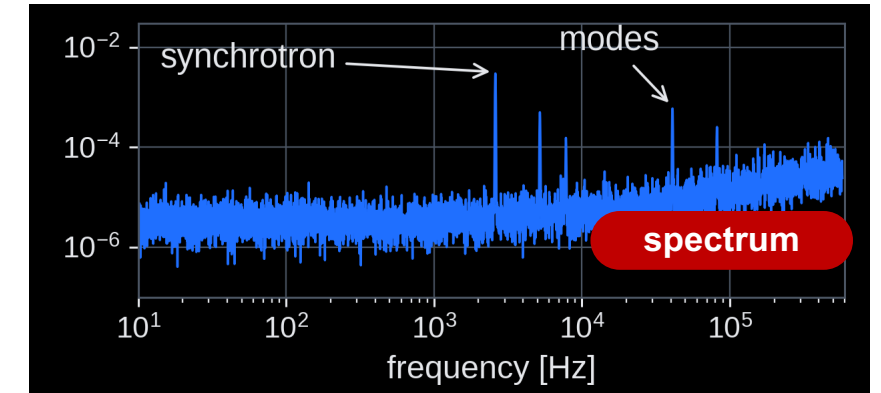
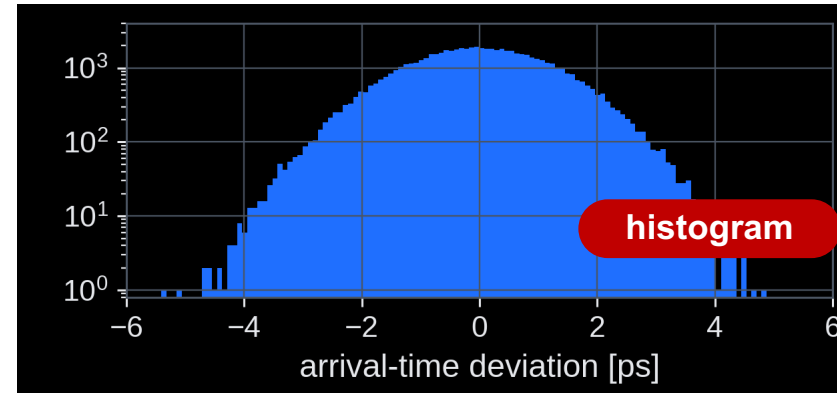
You know how to measure a bunch's jitter: we measure our packets the same way.

Electron bunches *sketch*

pickup



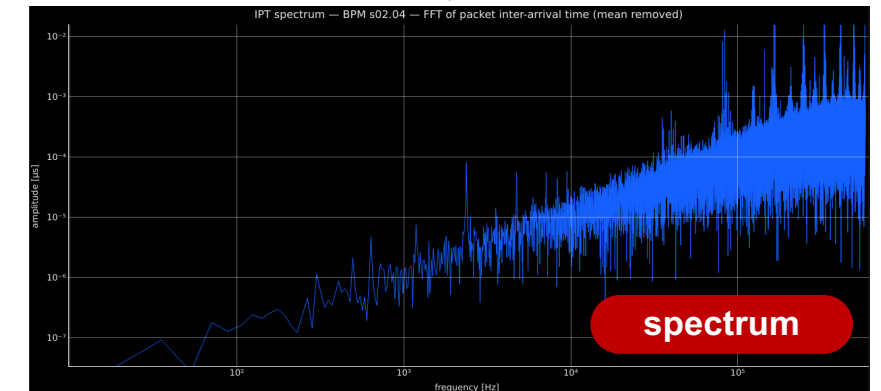
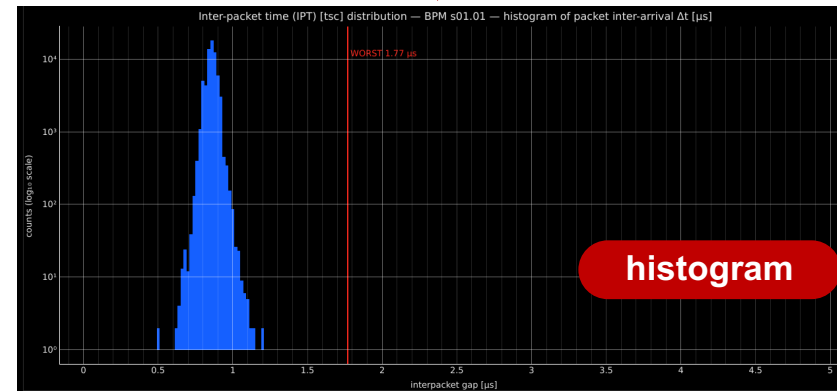
one bunch per bucket, every turn



UDP packets *measured, fast-01*



one packet per BPM, every turn



Packets as bunches: transport and acquisition layers

Transport jitter

- BPM FPGA
- Switch
- Network card
- Queuing and reordering on the link

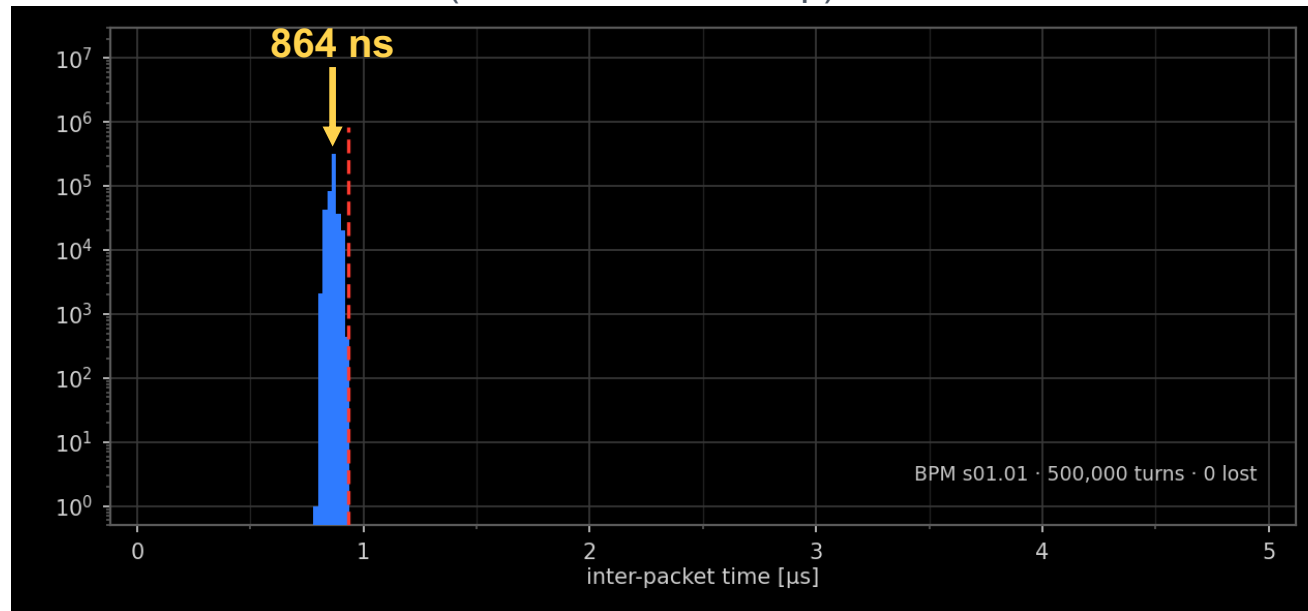
99.999 % of the packets within 0.80–0.93 μ s

Acquisition jitter

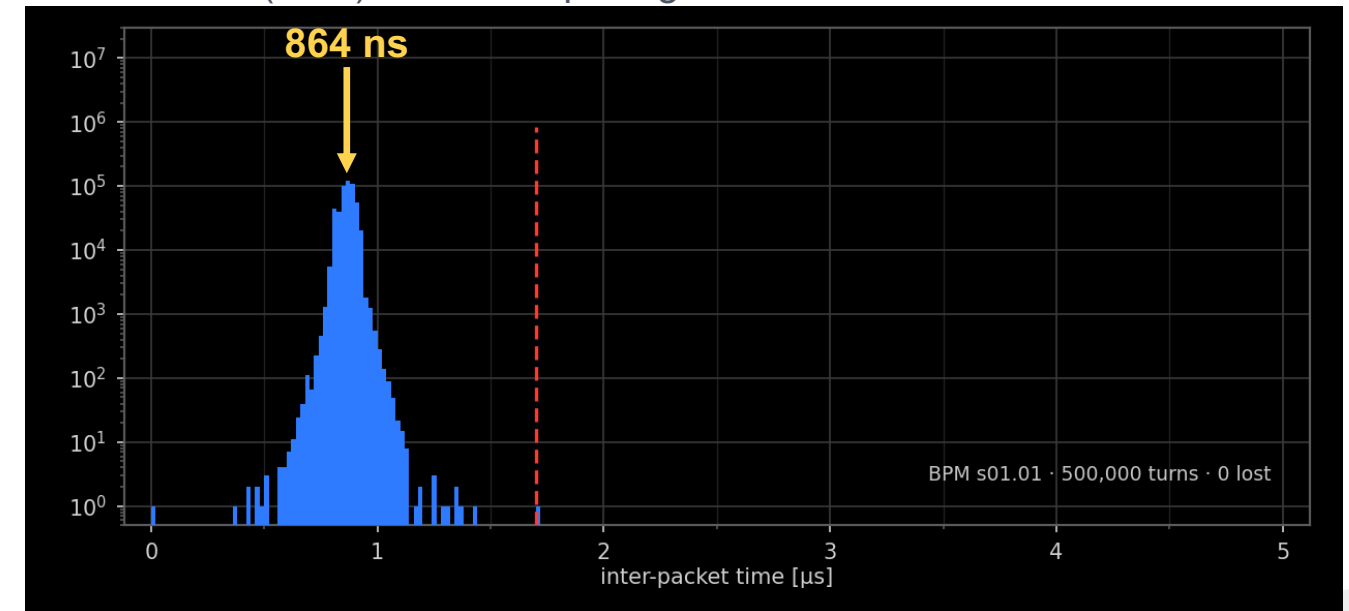
- DMA, PCIe, IO fabric
- Acquisition software on the polling core

99.999 % of the packets within 0.42–1.37 μ s

Network Card clock (hardware timestamp): on arrival

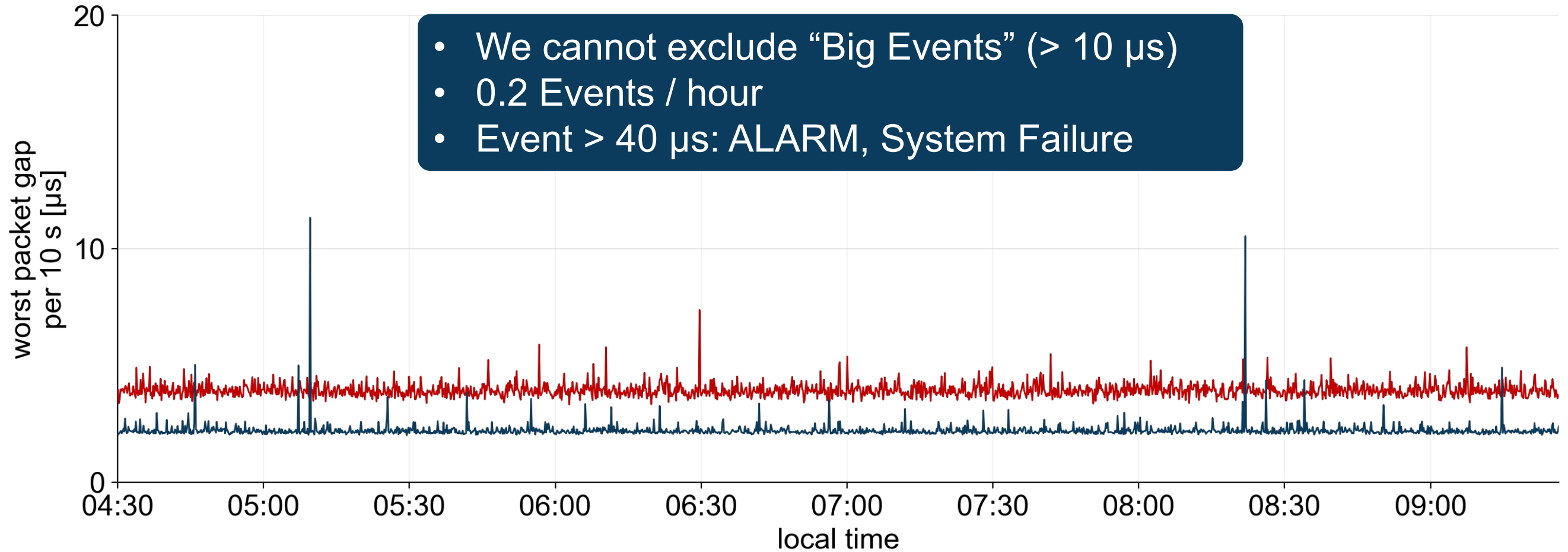


CPU clock (TSC): when the polling core sees it



The worst case jitter every 10 s, on both servers, tuning/kernel patching still ongoing

- We cannot exclude “Big Events” ($> 10 \mu\text{s}$)
- 0.2 Events / hour
- Event $> 40 \mu\text{s}$: ALARM, System Failure



- fast-02 · 195 Mpps in, 195 Mpps out · loopback · median $3.9 \mu\text{s}$, max $7.4 \mu\text{s}$
- fast-01 · 97 Mpps in · real BPMs · median $2.2 \mu\text{s}$, max $11.3 \mu\text{s}$

CRUDA (1): Control Real-time Ultra Direct Access

An inter-core light-weight control system

Applications on their cores

BPM

X, Y of every BPM, every turn

LLRF

phase and amplitude of every cavity

Power supplies

set and read the currents

The CRUDA table

register	name, as in Tango	address	value
	sr/diagnostics/rtbpm_s01.01/X	0x7f3a00001000	+12.4 μm
	sr/diagnostics/rtbpm_s01.01/Y	0x7f3a00001008	−3.1 μm
	sr/llrf/llrf_s03.01/Phase	0x7f3a00042000	12.37°
	sr/llrf/llrf_s03.01/Amplitude	0x7f3a00042008	0.982
	sr/power_supply/ch_s01.01/Current	0x7f3a00080000	1.250 A

... one row for every value of the machine

Each application registers its values once: the label name (the same as Tango) and the address.

Then a read is one load and a write is one store, core to core: 45–260 ns.

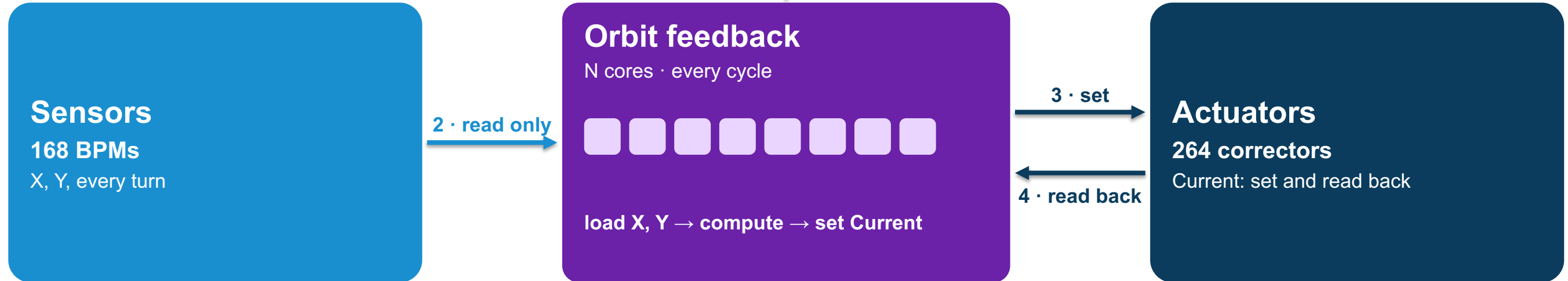
CRUDA (2): the orbit feedback, built on names

1 · At start-up

the feedback asks the CRUDA table for the offsets of all its names

`sr/diagnostics/rtbpm_*/X, Y`
336 positions · read

`sr/power_supply/ch_*/Current`
264 x 2 currents · set and read



The feedback is a generic process, configured by choosing among all the sensors and actuators available in CRUDA.

Basic algorithm: weighted sensors and actuators, SVD inversion with Tikhonov regularization, PID.
Ready to host more complex algorithms.

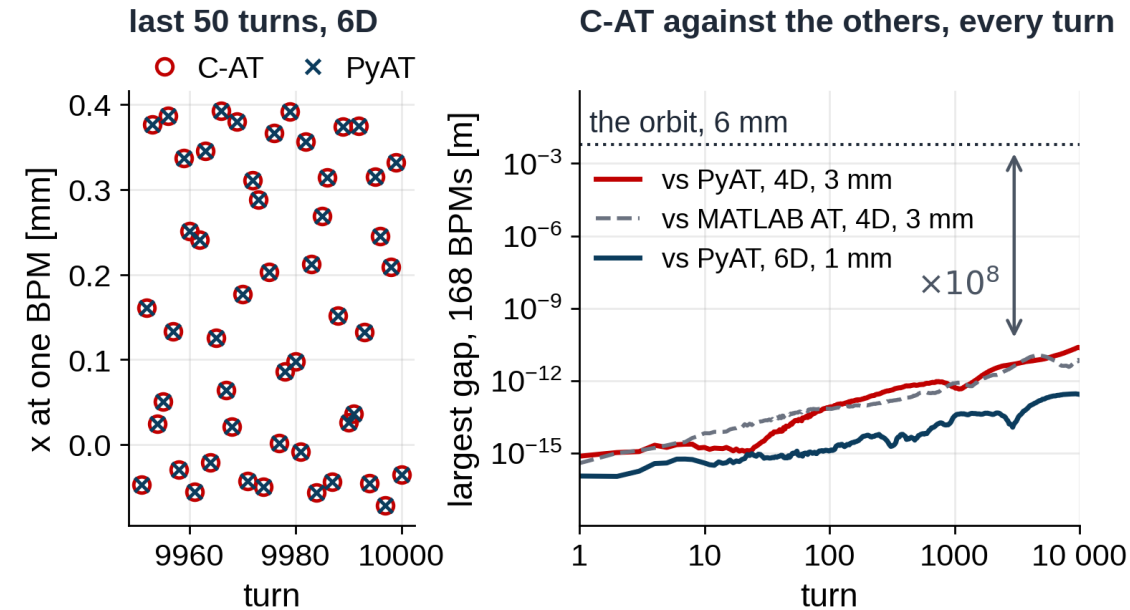
C-AT (1): a C simulator on the front-end, made for the feedbacks

Reimplement AT linear optics

~25 Hz

~2600 elements, one model cycle every 40 ms: more than the 10 Hz needed to follow the undulator gaps

Checked against the reference AT, PyAT



One particle, 10,000 turns, 168 BPMs: largest gap to PyAT 2.4×10^{-11} m. β at all 2575 elements: within 1×10^{-5} m of MATLAB-AT.

A narrower lens, built for operations

- Linear optics: orbit, β , μ , η , tunes
- Estimate orbit between the BPMs
- Calculate theoretical response matrices
- Evaluate cross-talk between feedbacks

Not a ring-design code

Same physics as AT, a narrower lens, in C, run multiple instances without loading the control system.

C-AT (2): the twin, a Tango server between two worlds

From the lattice file to attributes names

element · parameter	AT field
Q Quadrupole strength K	PolynomB[1]
S Sextupole strength K2	PolynomB[2]
C Corrector kick	KickAngle
M Any magnet mechanical offset, roll	T1, T2, R1, R2
RF RF cavity voltage, frequency	Voltage, Frequency
B BPM orbit x, y, computed	from the optics

One name per parameter: `SR__MAGNET__QF_S01.01__strength`

~9,000 names, generated at start-up; each with an error model (gain, offset, noise, history)

Tango · high level control room, scripts, archive: by name, in ms

knobs, setpoints ↓ optics, history ↑

C-AT digital twin · one device server

every ~40 ms (~25 Hz): clone the lattice, compute in parallel, publish atomically

atlinopt6 orbit, β , μ , η , tunes

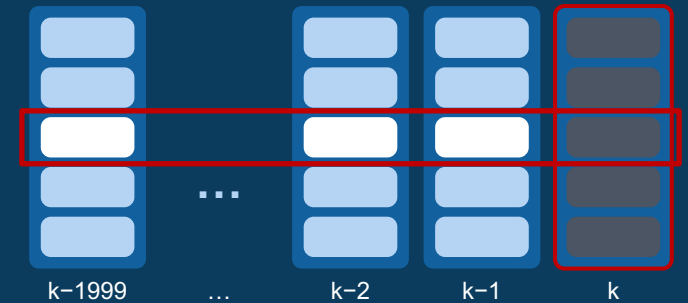
atchromaticity ξ_x , ξ_y

atohmienvelope σ , emittance

optimization feedbacks, LOCO, matching

linear modeling response matrices

history: a ring of 2000 lattices, ~10 GB



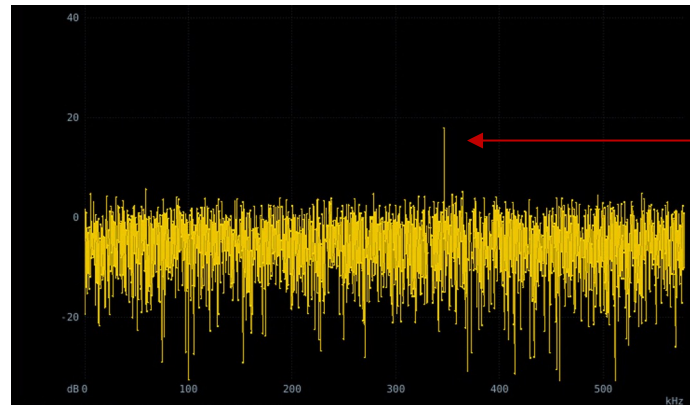
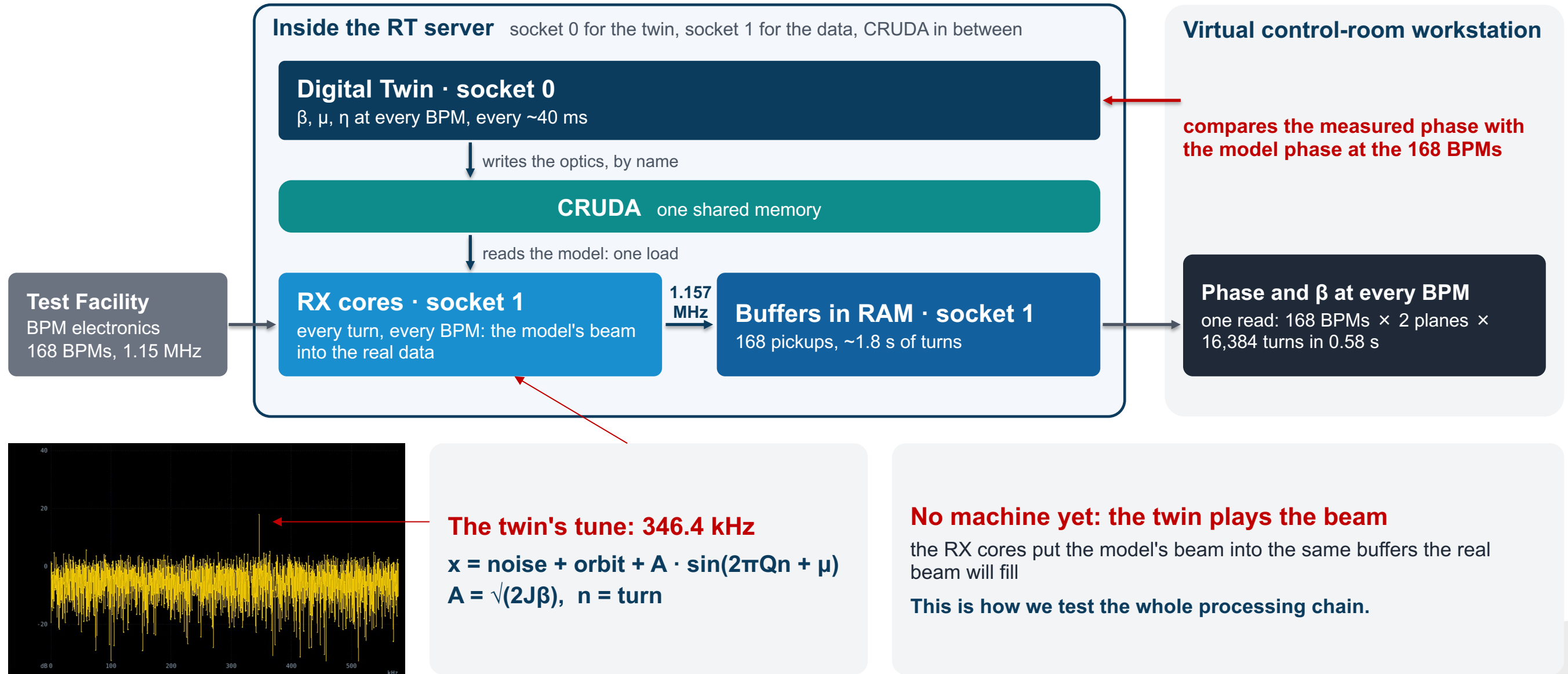
model for the feedbacks ↓ machine data from RAM ↑

CRUDA · low level RT cores: BPMs, power supplies, feedbacks · one load, one store

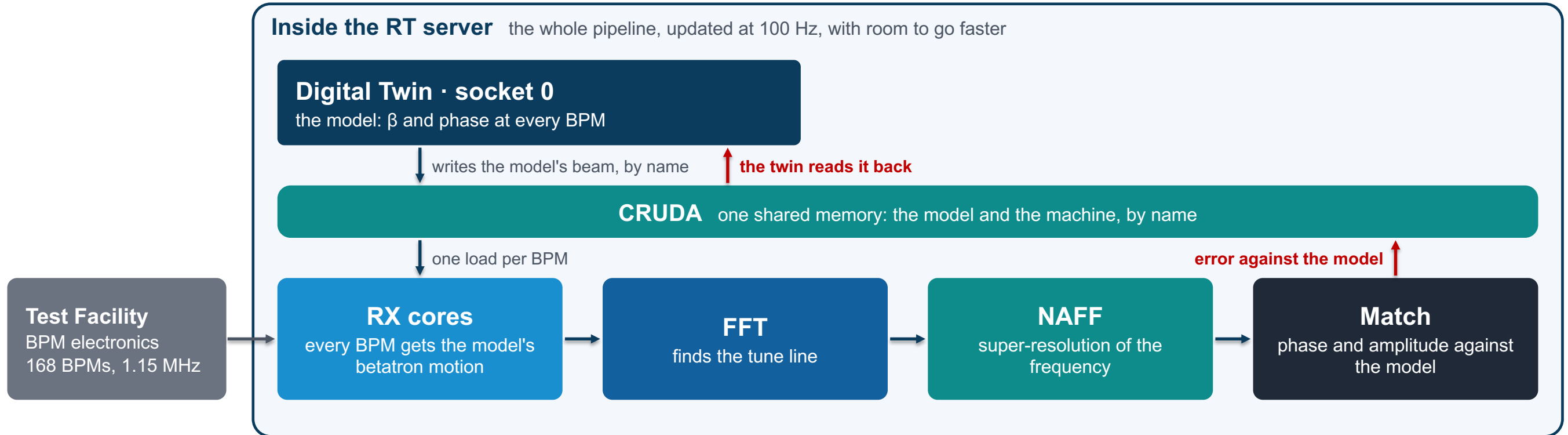
Every lattice parameter has one name, in both worlds: Tango for people, CRUDA for the RT cores.

The twin sits in between: knobs from above, the machine from below, the model served to both every ~40 ms, with its history.

Test the framework (1): the twin and the BPM data on the same server



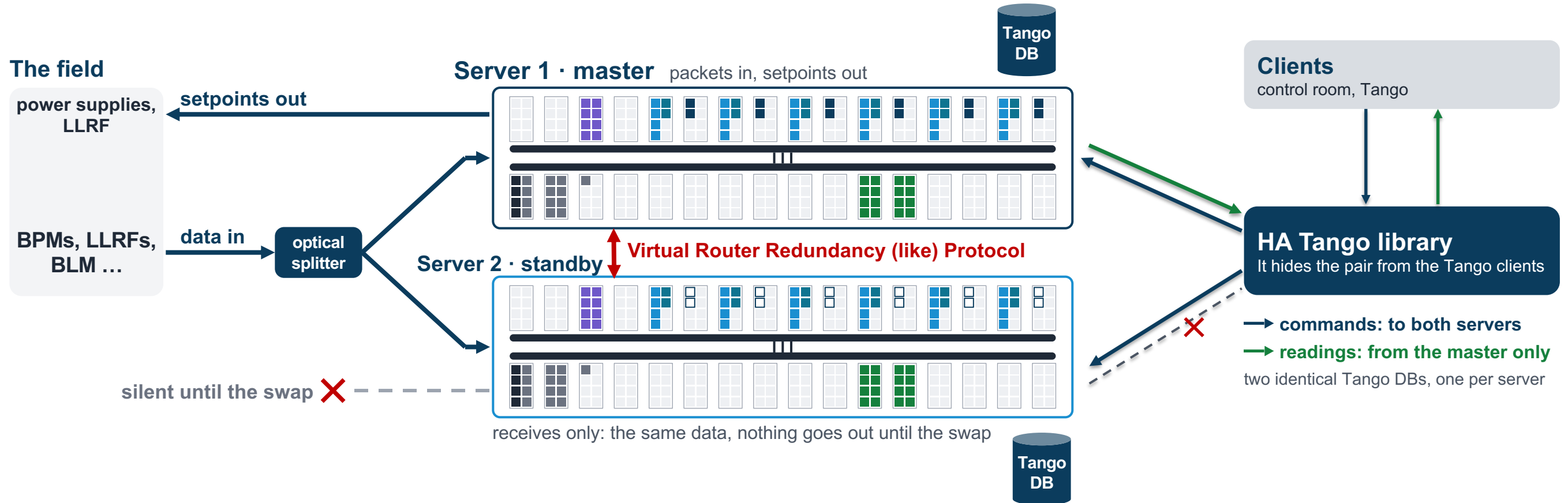
Test the framework (2): the whole loop inside the server



The whole loop closes inside one server, 100 times a second.

Next: the real beam. Each BPM's signal-to-noise, read from its own spectrum, sets its error bar on phase and β .

What comes next: High Availability



Not from day zero.

HA-Tango breaks binary compatibility with Tango. Each server can still be reached directly with plain Tango.

Conclusions: a unique moment

1 Phenomenal hardware

512 HT cores · 1.5 TB ·
600 Gbit/s

one commodity server takes the fast data of the whole machine

2 AI helps

Fewer bugs, less code, fewer layers

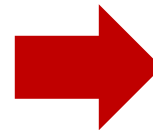
3 Ready to ingest data

every turn, in RAM

data-driven next to model-driven: the twin and the data in one memory

Post-mortem buffers

triggered, limited depth, one shot, after the fact



Streaming

every turn, every BPM, always on: the analysis runs while it happens

Let your bare metal stream.

