Elettra Sincrotrone Trieste

DEVELOPMENT OF ETHERNET BASED REAL-TIME APPLICATIONS IN LINUX USING DPDK

MOPHA044

In the last decade Ethernet has become the most popular way to interface hardware devices and instruments to the control system. Lower cost per connection, reuse of existing network infrastructures, very high data rates, good noise rejection over long cables and finally an easier maintainability of the software in the long term are the main reasons of its success. In addition, the need of low latency systems of the High Frequency Trading community has boosted the development of new strategies, such as CPU isolation, to run real-time applications in plain Linux with a determinism of the order of microseconds. DPDK (Data Plane Development Kit), an open source software solution mainly sponsored by Intel, addresses the request of high determinism over Ethernet by bypassing the network stack of Linux and providing a more friendly framework to develop tasks which are even able to saturate a 100 Gbit connection. Benchmarks regarding the real-time performance and preliminary results of employing DPDK in the acquisition of beam position monitors for the fast orbit feedback of the Elettra storage ring will be presented.

DPDK DATA PLANE DEVELOPMENT KIT

Gold Members

1 - Motivation

The Linux network stack is unsuitable for developing time sensitive network applications.

•The POSIX socket operations (system calls), which transfer control from the application layer to the kernel have significant overheads (e.g. **data copy, context switch and CPU cache pollution**).

•The **network performance has grown faster than the one of the CPUs** due the stagnation in the single thread performance.

2 - Bypassing Linux network stack

Software solutions provided by:

•**HW vendors**: VMA by Mellanox, OpenLoad by SolarFlare/Xilinx

•**Open source**: NetMap, PF_Ring, DPDK.

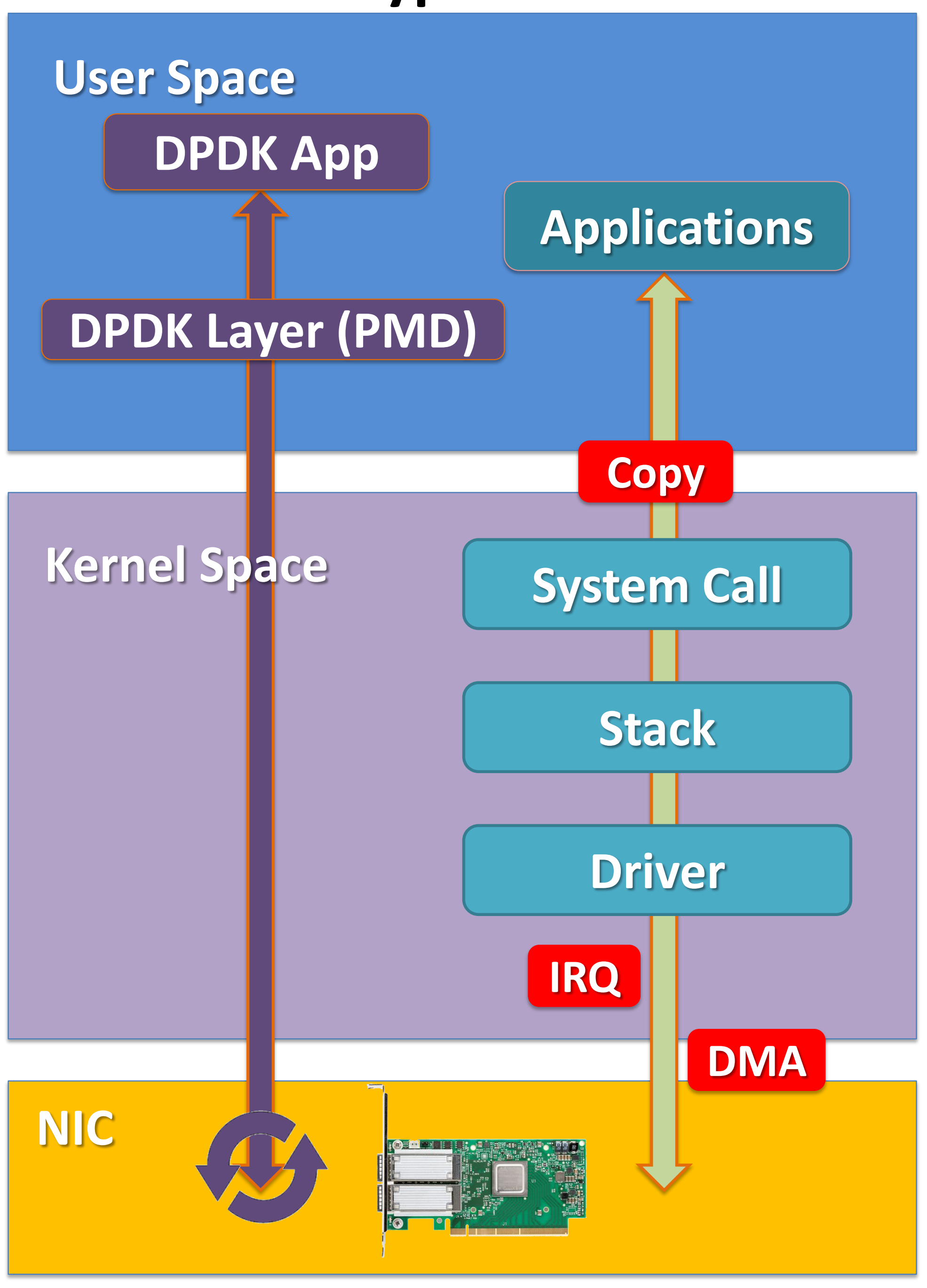
3 - DPDK description

DPDK, initially developed by **Intel** in 2010, supported directly by the Linux Foundation and sponsored by market leaders as **ARM**, **Red Hat**, **AT&T** and **Ericsson.**

•The **user has to assign to DPDK a pool of network interface cards (NIC) and CPU cores** which become unavailable to the Linux kernel.

•**User code spins (PMD) on a dedicated core waiting for incoming packets.** Once the packets are available, data processing and retransmission can be managed by the same core or executed on other reserved cores.

DPDK Bypass Scheme



4 - Official test reports (no information on jitter)

Achieved 100Gbit line rate in roundtrip configuration between Mellanox ConnectX5 and a Traffic Generator (Ixia) – 12 CPU cores – 148 Mpss (64 bytes UDP packet). Mpss=milions of packets per second

5 – Measuring jitter (HW setup)

At the Elettra synchrotron light source **the most jitter-sensitive real-time application** based on Ethernet is the **fast orbit feedback**:

•96 Beam position monitors (BPM) transmitting UDP packets at 10kHz

•Twelve VME-PowerPC systems receiving data, performing calculation and setting 164 magnet power supplies

Duplicated and forwarded 1 milion of UDP packets per sec. to a single rackmount server (dual socket Xeon 2637) by means of 4 10Gbit fibre optic links (4 10GbE Intel X710 ports).

6 – Measuring jitter (SW setup)

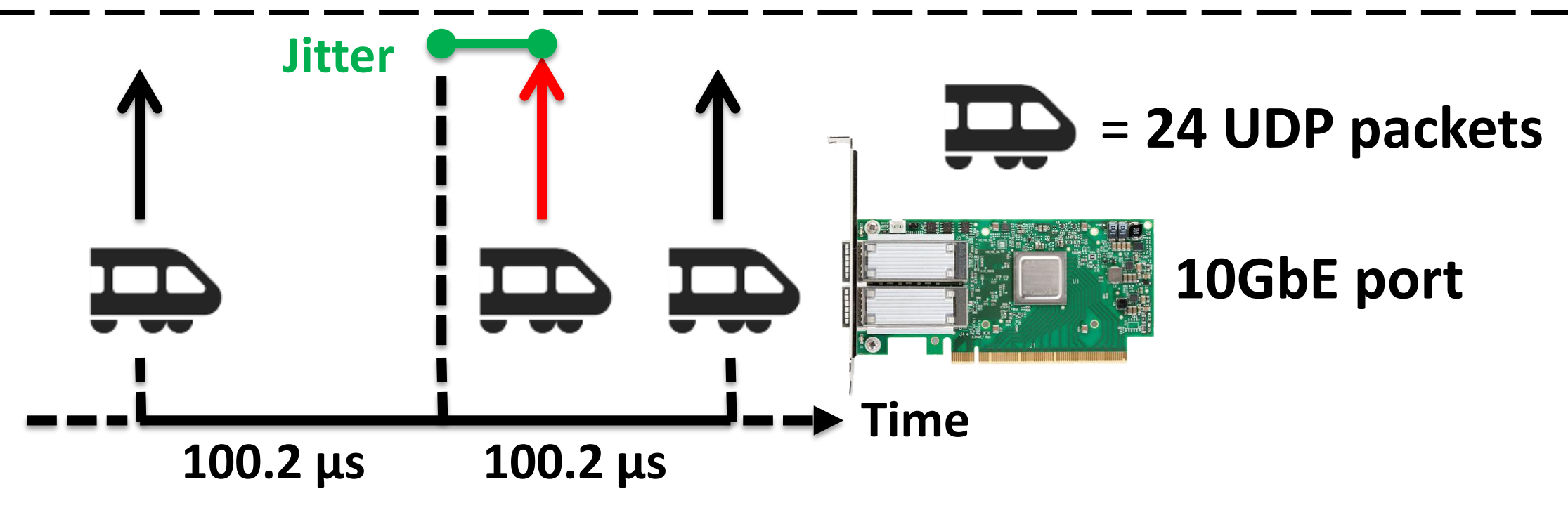
•Linux Ubuntu 18.04 + Kernel 4.15.0 + DPDK 18.04

•Optimized BIOS and Linux for max performance (no RT patch)

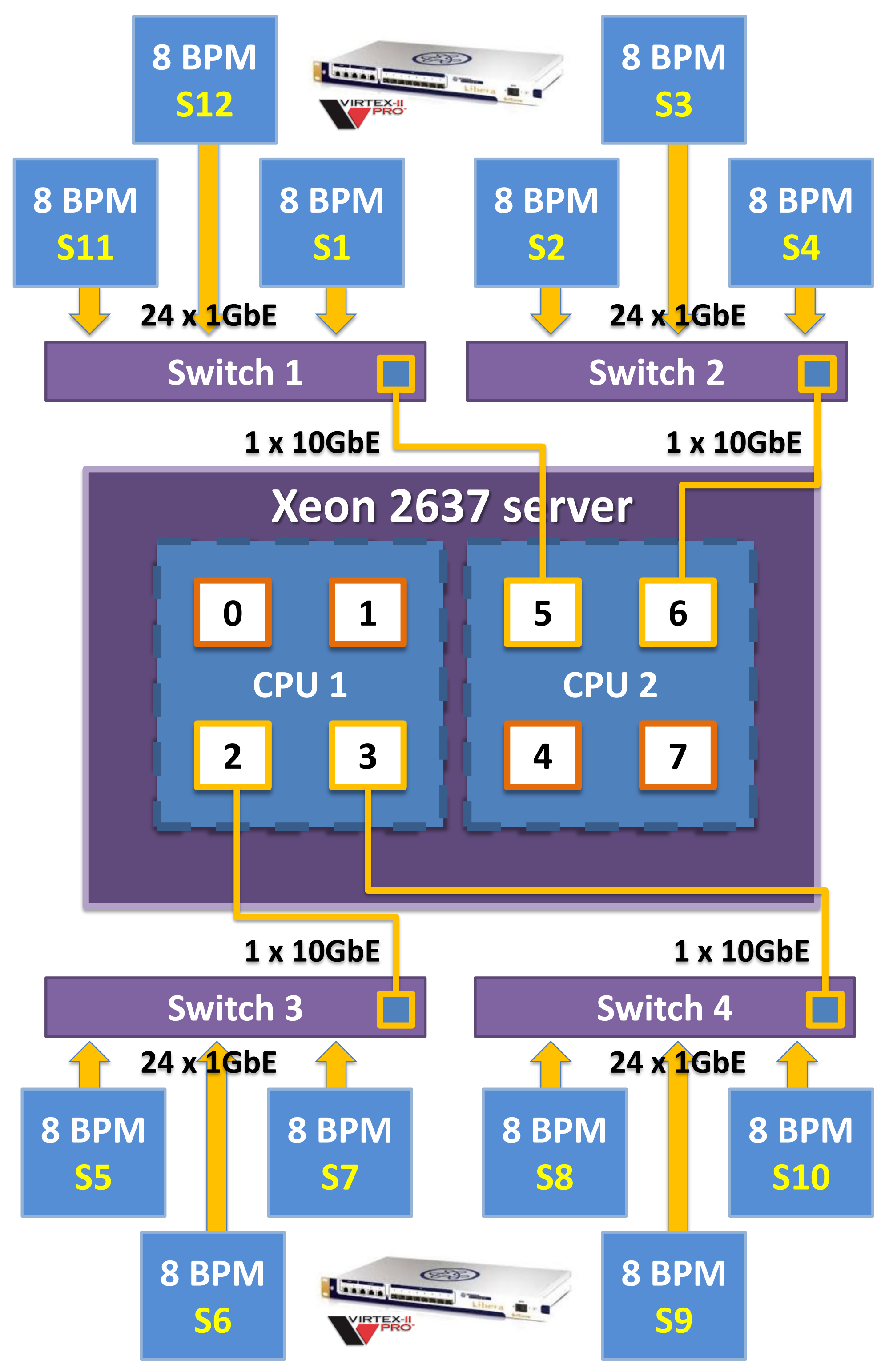
•**CPU core 0** reserved to Linux loaded with *"stress"*

•**CPU cores 2,3,5,6** reserved to four user space DPDK processes to acquire each one 10GbE port

The difference between the feedback repetition period and the measured time between the arrival of two consecutive bunches of 24 packets (belonging to one fourth of the BPMs) on a single 10GbE port is a realistic estimation of the jitter per cycle of the system



Data flow during DPDK test



7 – DPDK test results

Jitter	Core 2	Core 3	Core 5	Core 6
< 1 µs	0%	0%	0%	0%
< 2 µs	99.91%	99.88%	99.98%	99.98%
< 3 µs	0.049%	0.05%	0.015%	0.013%
< 4 µs	0%	0%	0%	0%
< 5 µs	0.032%	0.0598%	0.0018%	0.0023%
< 10 µs	0.026%	0.0038%	0.0004%	0.0006%
< 20 µs	0%	0%	0%	0%
< 30 µs	0%	0.0006%	0.0002%	0.0002%
<40 µs	0.006%	0%	0%	0%
> 40 µs	0%	0%	0%	0%

•In 99.95% of cycles jitter less then 3 µs

•Worst case < 40 µs (cache misses)

•Estimated max. throughput: 33 Mpss -> 8 Mpss per core

POSIX socket connection running on isolated core 2 acquiring one 10 GbE port:

•88 % of packets received in less then 10 µs

•10 % of the packets received in less then 100 µs

•Worst case: 12.1 ms

G. Gaio*, G. Scalamera, Elettra-Sincrotrone Trieste S.C.p.A., Trieste, Italy
* giulio.gaio@elettra.eu