areaDetector EPICSv4 modules

Bruno Martins

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

areaDetector EPICSv4 modules

Bruno Martins

Goal

Process the huge amount of data generated by recent detectors in **real time**

Motivation

- Eiger 1M: 1030x1065 @ 3 kHz
- Eiger 4M: 2070x2167 @ 750 Hz
- Eiger 9M: 3110x3269 @ 238 Hz
- Eiger 16M: 4150x4371 @ 133 Hz

- All of them saturate a 10Gbps link:
  - That's a lot of data!

areaDetector v4 modules

- Comprised of:
  - A plugin, **NDPluginPva**: v4 server
  - A driver, **pvaDriver**: v4 client
- Developed in-house independently of other solutions
- Merged into areaDetector's ADCore on branch **`v4-plugin`**
  - http://github.com/areaDetector/ADCore

Plugin: NDPluginPva

- In **`processCallbacks`**:
  - Receives an **`NDArray`**;
  - Zero-copies it into an **`NTNDArray`** (creates a **`shared_vector`** with underlying data pointing to **`NDArray`**'s **`pData`**);
  - Publishes the **`NTNDArray`** as a PV using a pvDatabase instance;

Driver: pvaDriver

- In **`monitorEvent`**:
  - Receives an **`NTNDArray`**;
  - Copies it into an **`NDArray`**;
  - Publishes the **`NDArray`** to driver's listeners: **`doCallbacksGenericPointer()`**;

- No Zero Copy yet – see next slide.

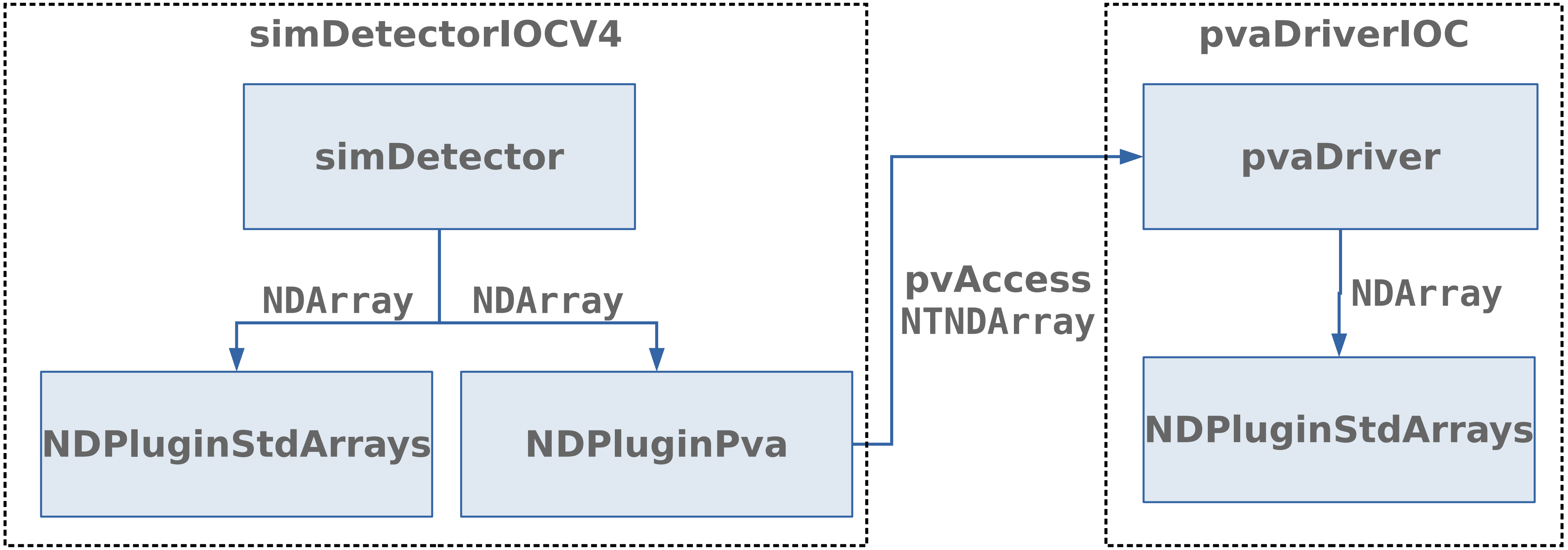
Zero Copy

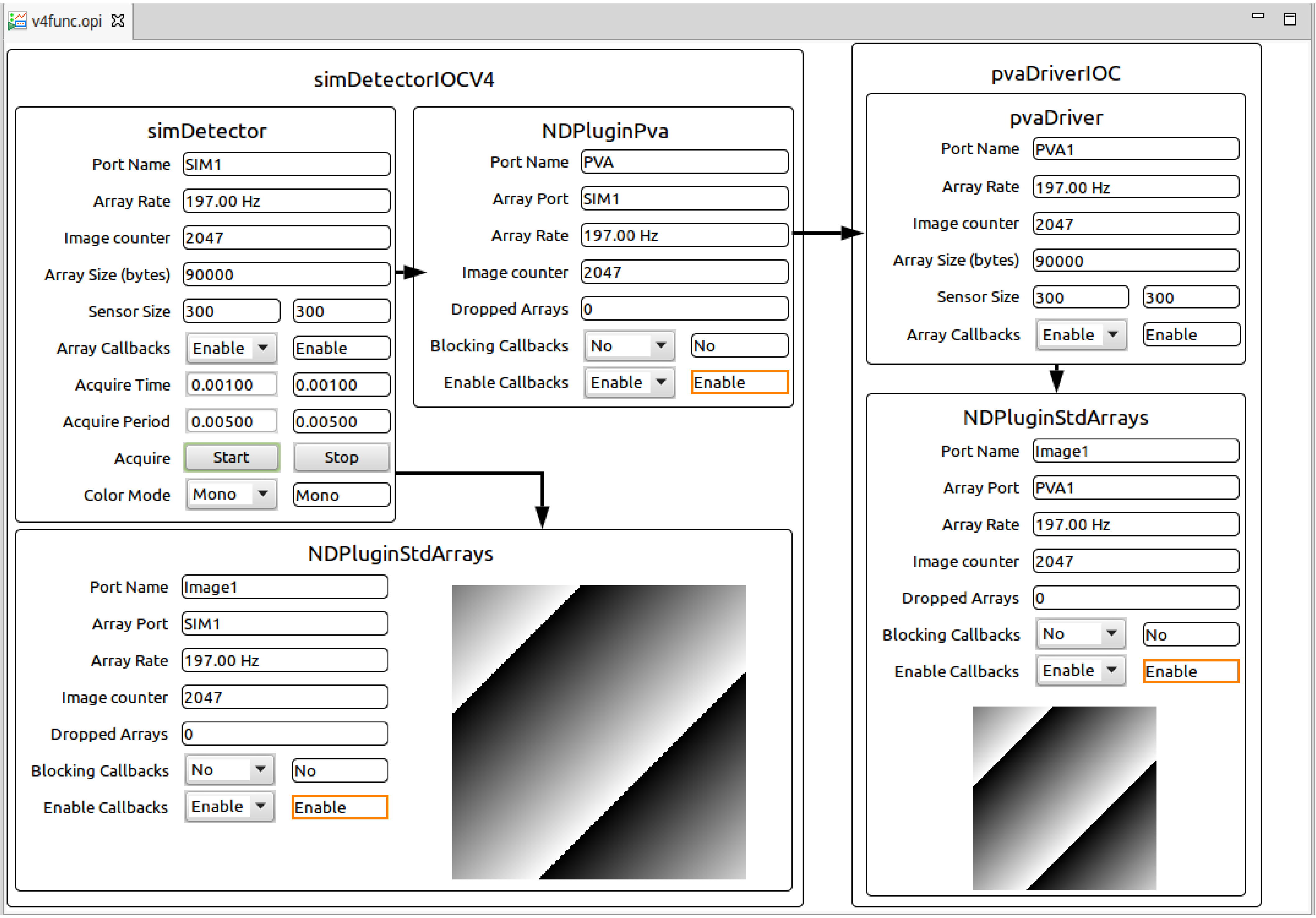
- **`NDArray`** → **`NTNDArray`** *works fine*
  - Underlying **`shared_vector`** is smart: can be told to **`release()`** the original **`NDArray`** in its destructor;

- **`NTNDArray`** → **`NDArray`** *not so much*
  - **`NDArray`** can be allocated pointing to **`NTNDArray`**'s **`shared_vector`**'s data, but smart pointer has to be kept for the lifetime of the **`NDArray`**. However:
  - No current way to make **`NDArray`**'s **`release()`** dispose of the smart pointer;
  - Driver can keep the smart pointer, but for how long? How to know **`NDArray`** that was passed to the plugins is no longer being used?

Test 1: Functionality

- **Question: Do they work?**
- Both IOC's on the same computer





simDetectorIOCV4

simDetector

Port Name	SIM1	
Array Rate	197.00 Hz	
Image counter	2047	
Array Size (bytes)	90000	
Sensor Size	300	300
Array Callbacks	Enable	Enable
Acquire Time	0.00100	0.00100
Acquire Period	0.00500	0.00500
Acquire	Start	Stop
Color Mode	Mono	Mono

NDPluginPva

Port Name	PVA	
Array Port	SIM1	
Array Rate	197.00 Hz	
Image counter	2047	
Dropped Arrays	0	
Blocking Callbacks	No	No
Enable Callbacks	Enable	Enable

NDPluginStdArrays

Port Name	Image1	
Array Port	SIM1	
Array Rate	197.00 Hz	
Image counter	2047	
Dropped Arrays	0	
Blocking Callbacks	No	No
Enable Callbacks	Enable	Enable

pvaDriverIOC

pvaDriver

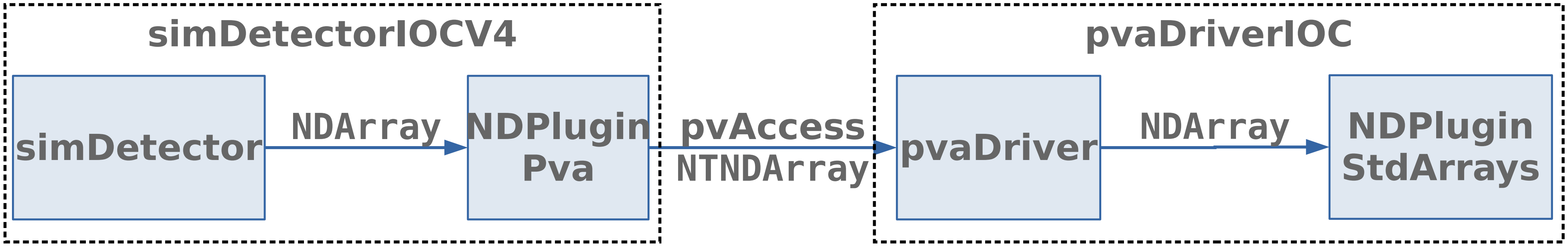
Port Name	PVA1	
Array Rate	197.00 Hz	
Image counter	2047	
Array Size (bytes)	90000	
Sensor Size	300	300
Array Callbacks	Enable	Enable

NDPluginStdArrays

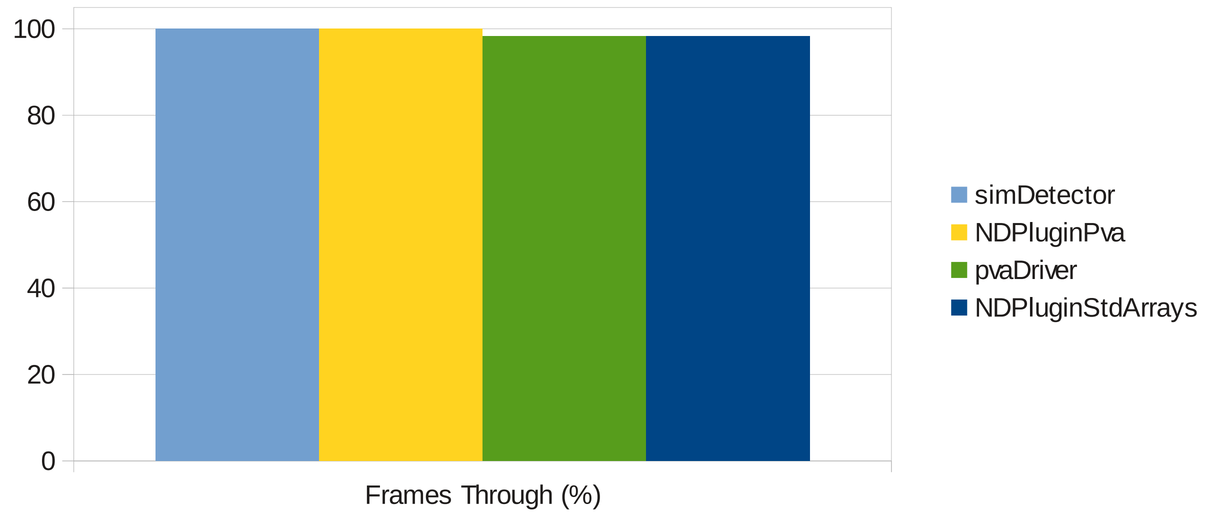
Port Name	Image1	
Array Port	PVA1	
Array Rate	197.00 Hz	
Image counter	2047	
Dropped Arrays	0	
Blocking Callbacks	No	No
Enable Callbacks	Enable	Enable

Test 2: Performance

- **Question: can they handle more than 10 Gbps?**
- Both IOC's on the same computer
- **simDetector**:
  - 5000x5000 @ 60 Hz: little over 11 Gbps
  - ImageMode: Multiple, NumImages: 10000
- Both plugins with non-blocking callbacks.
- Results are the avearage of 10 runs.

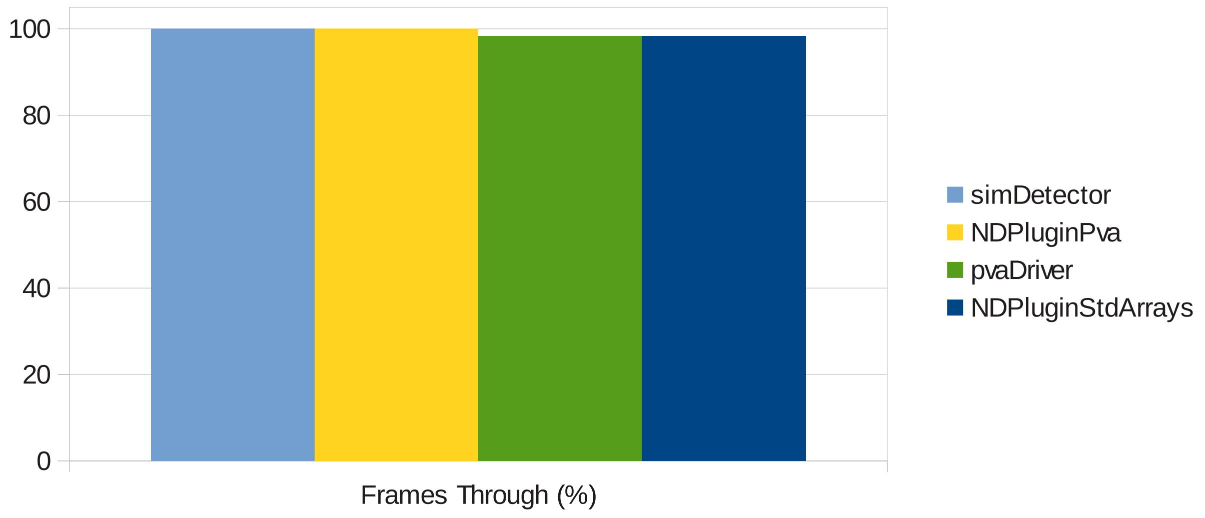


Test 2: Performance – produced frames



	Frames Lost (Avg)	Frames Through (Avg)	Frames Through (Avg %)
simDetector	0	10000	100
NDPluginPva	0	10000	100
pvaDriver	171.3	9828.7	98.287
NDPluginStdArrays	173.3	9826.7	98.267

Test 2: Performance – produced frames

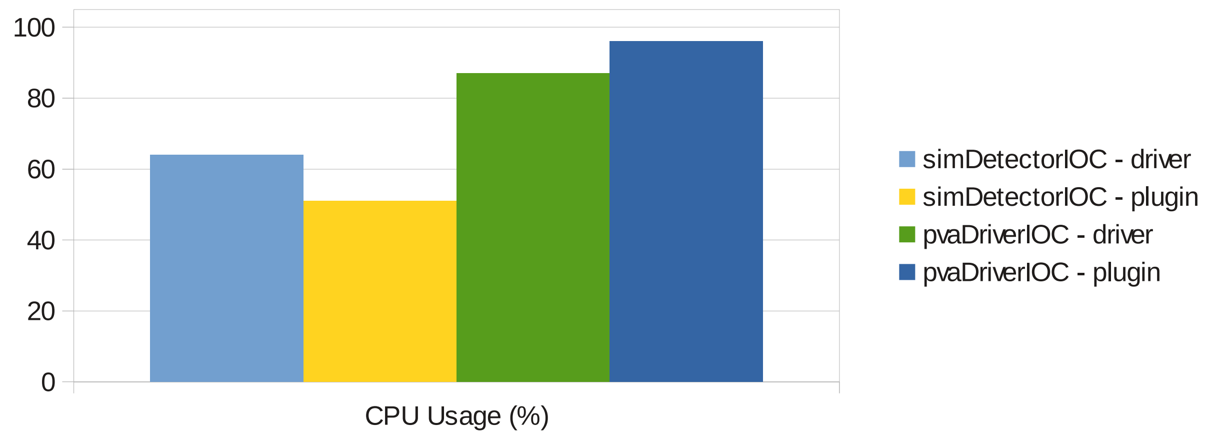


- **NDPluginPva** never lost frames.
  - Zero Copy makes it really fast.
- It's worth noting that even **without** zero copy **pvaDriver** lost only **2%** of the frames.

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

Test 2: Performance – CPU usage / thread



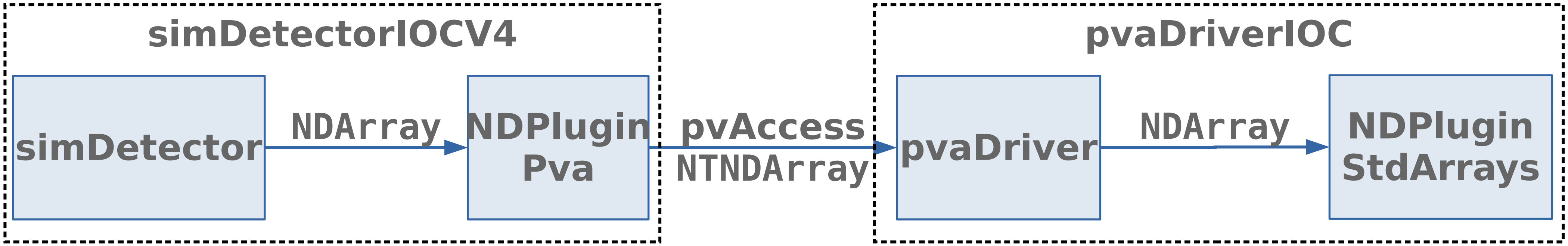
	CPU Usage
simDetectorIOC - driver	~64%
simDetectorIOC - plugin	~51%
pvaDriverIOC - driver	~87%
pvaDriverIOC - plugin	~96%

Test 2: Performance - Computer Specs

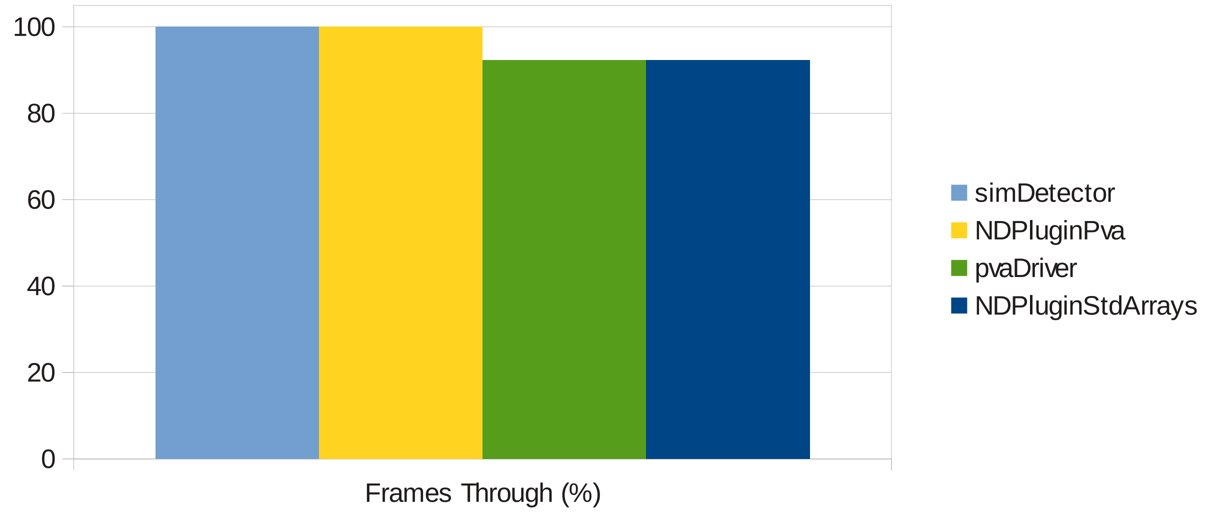
- Intel Xeon E5-2643, 24 cores @ 3.40 GHz
- 256GB RAM
- Debian Wheezy 7.8 64-bit

Test 3: Transfer between computers

- **Question: can they saturate a *real* 10Gbps link?**
- IOC's on *different* computers
- **simDetector**:
  - 5000x5000 @ 50 Hz: ~10 Gbps
  - ImageMode: Multiple, NumImages: 10000
- Both plugins with non-blocking callbacks.
- Results are the avearage of 5 runs.

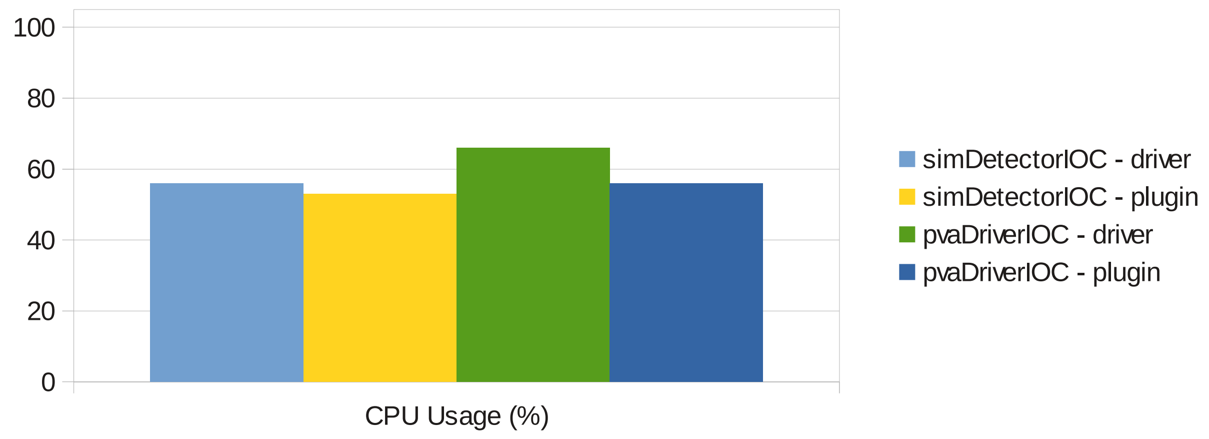


Test 3: Transfer – produced frames



	Frames Lost (Avg)	Frames Through (Avg)	Frames Through (Avg %)
simDetector	0	10000	100
NDPluginPva	0	10000	100
pvaDriver	773.6	9226.4	92.264
NDPluginStdArrays	773.6	9226.4	92.264

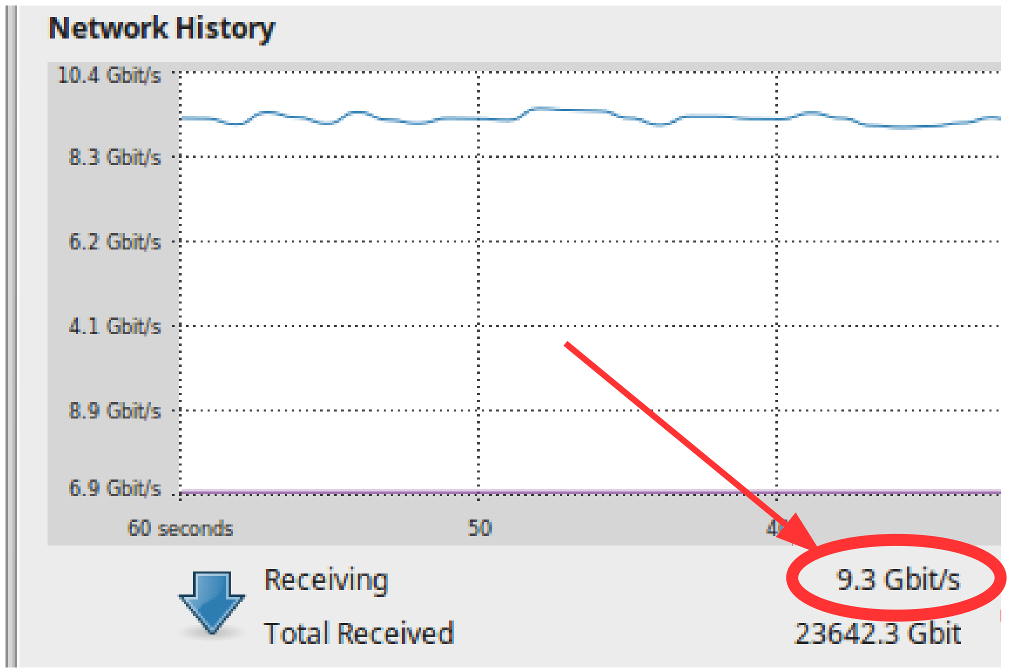
Test 3: Transfer – CPU usage / thread



	CPU Usage
simDetectorIOC - driver	~56%
simDetectorIOC - plugin	~53%
pvaDriverIOC - driver	~66%
pvaDriverIOC - plugin	~56%

Test 3: Transfer - bandwidth

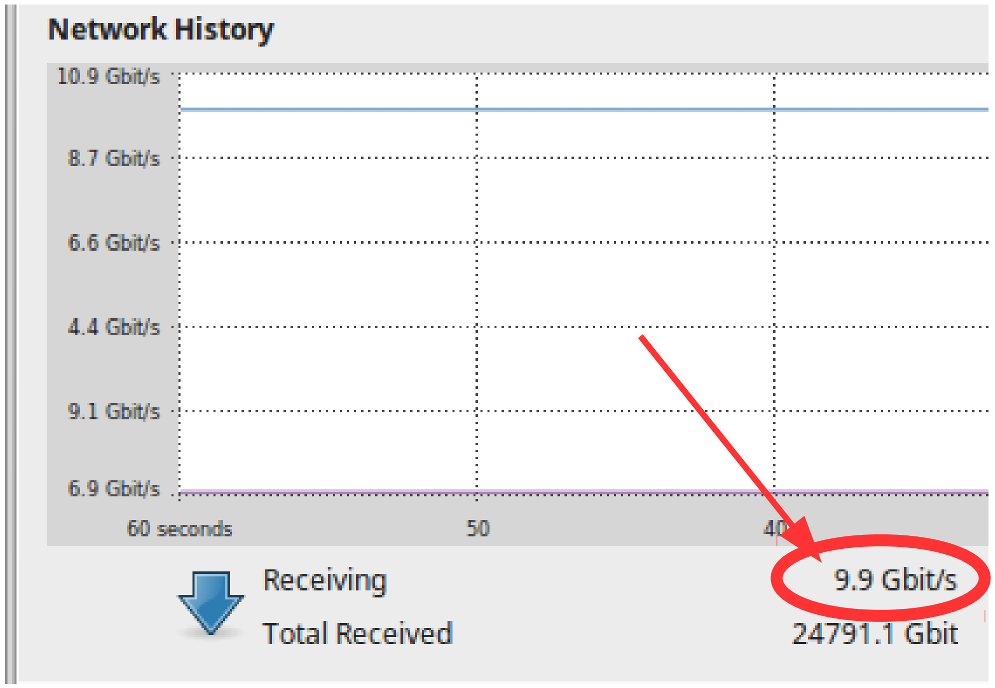
NDPvaPlugin + pvaDriver: **9.3 Gbps**



Receiving 9.3 Gbit/s

Total Received 23642.3 Gbit

Baseline (iperf network tool): **9.9 Gbps**



Receiving 9.9 Gbit/s

Total Received 24791.1 Gbit

Test 3: Transfer - Computer Specs

- simDetectorIOC:
  - Intel Xeon E5-2643, 24 cores @ 3.40 GHz
  - 256GB RAM
  - Debian Wheezy 7.8 64-bit

- pvaDriverIOC
  - Intel i7-4770, 8 cores @ 3.40 Ghz
  - 16GB RAM
  - Linux Mint 17.1 64-bit

Conclusion

- Plugin and server are ready to be used
  - Available on areaDetector's **`v4-plugin`** branch
- They have a high throughput
- They don't saturate the CPU
  - Although the CPU tested was powerful.

Future improvements

- Zero-copy on **pvaDriver**
  - Might depend on **`NDArray`** changes

- Better mechanism to detect frame losses by **pvaDriver**
  - 1 overrun might consist of more than 1 frame lost

- **NDPluginPva/pvaDriver** lossless mode:
  - Client tells the server to slow down if needed
    - How to handle multiple clients, then?
  - Depends on support from v4 protocol

Thank you!