Università degli Studi di Padova

Rate and bandwidth measurements

Antonio Bergnoli, Riccardo Brugnera, Vanessa Cerrone, Alberto Coppi, Alberto Garfagnini, Marco Grassi, Beatrice Jelmini, Ivano Lippi, **Andrea Serafini**, Andrea Triossi, **Riccardo Triozzi**, Katharina von Sturm

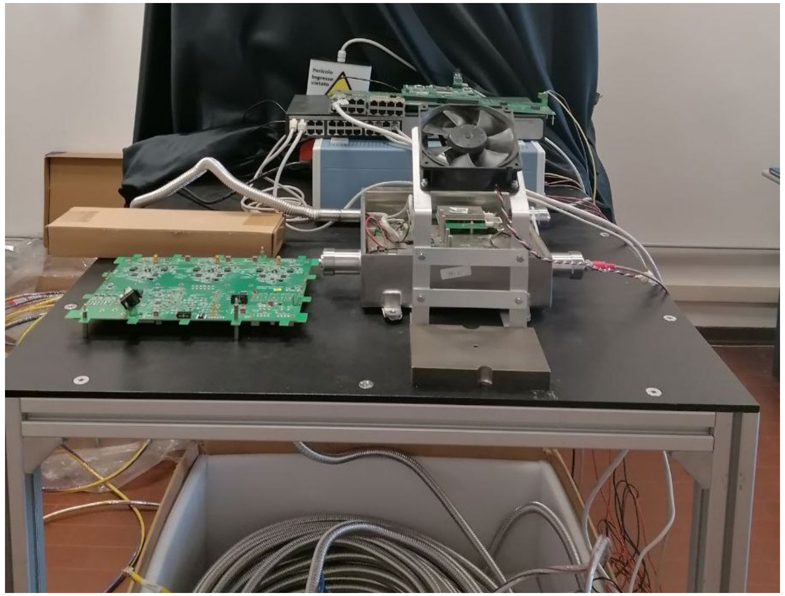
6 May 2022

Overview

1. IPbus and its implementation implementation

2. Data acquisition performances for a single GCU

3. Parallel acquisition with multiple GCUs



The IPbus suite

JUNO challenge: **acquire** and transfer **data** in parallel to the **remote monitoring** and control of the electronics.

IPbus suite of software and firmware implements a reliable high-performance control link specifically suited for particle physics electronics.

IPbus is a hardware and firmware solution that communicates over Ethernet using **UDP/IP** and consists of:

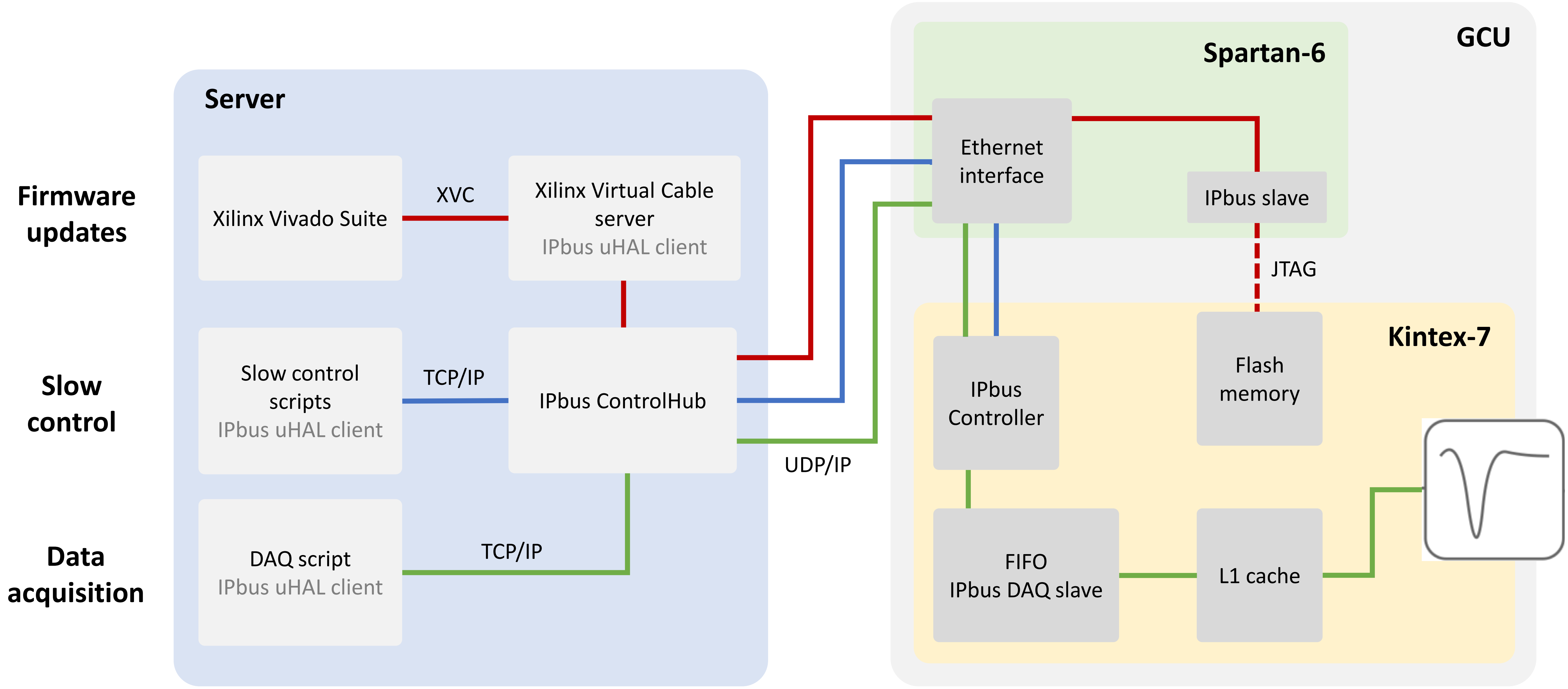
(i) A **firmware module** implementing the IPbus protocol within end-user hardware (e.g. JUNO's Kintex-7 and Spartan-6 FPGAs)

(ii) A **micro Hardware Access Library (uHAL)** providing an end-user C++/Python library for read/write operations on IPbus.

(iii) A **software application** called **ControlHub**, which mediates simultaneous hardware access from multiple uHAL clients.

While the UDP protocol does not include any native reliability mechanism, the use of **ControlHub** assures the duplication and re-ordering of any lost IPbus UDP packet, **providing a reliability mechanism** at software level.

IPbus implementation



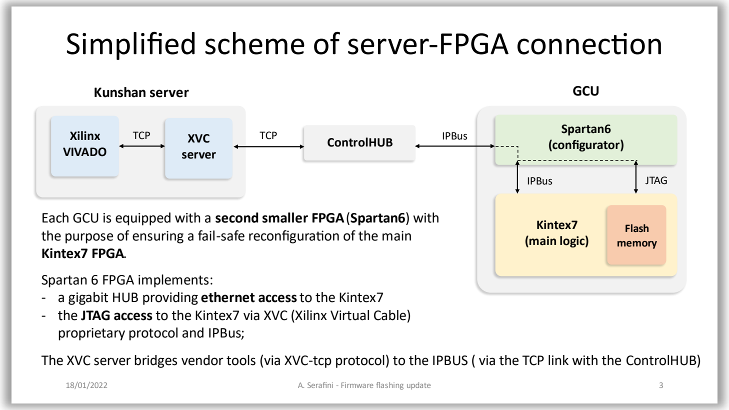
IPbus implementation

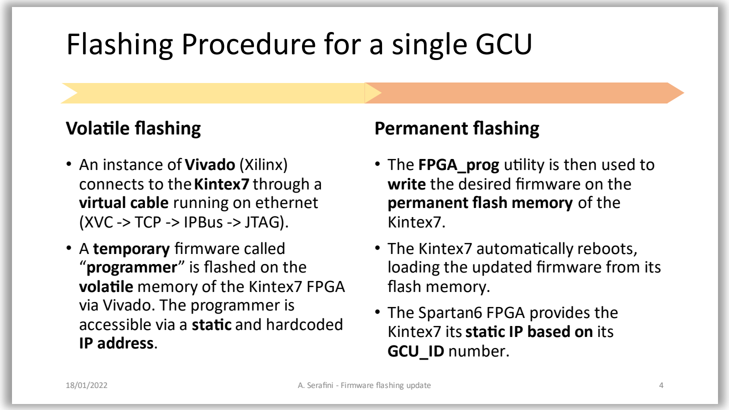
Firmware updates

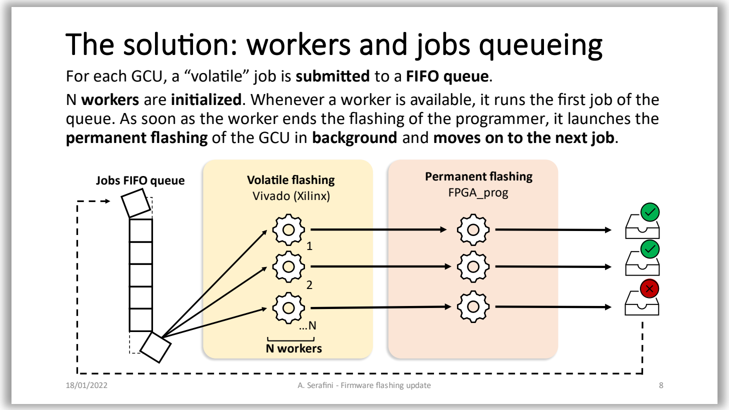
Slow control

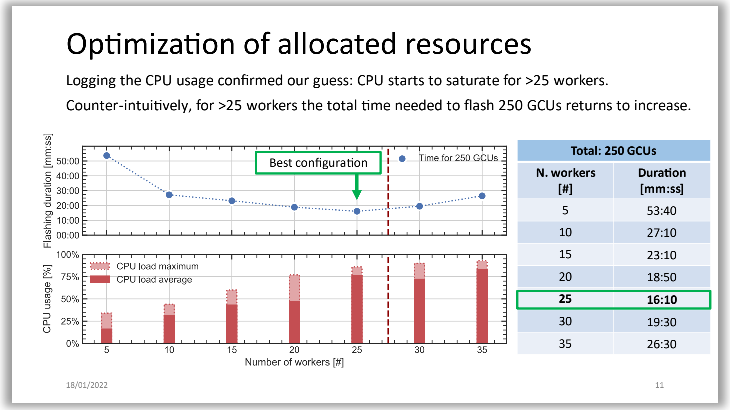
Data acquisition

My talk at Jan 2022 Collaboration Meeting (DocDB 7827)









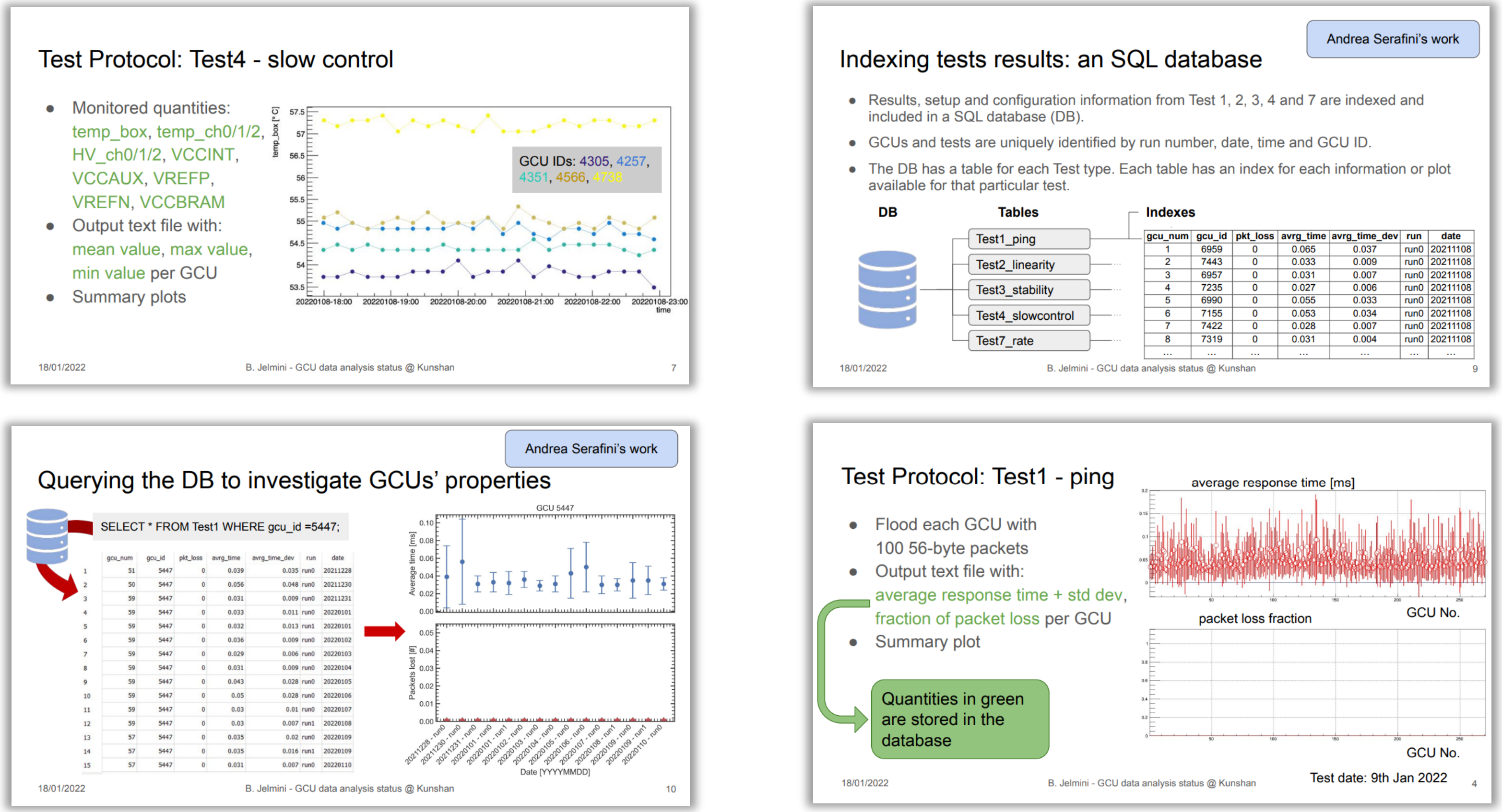
IPbus implementation

Firmware updates

Slow control

Data acquisition

Beatrice's talk at Jan 2022 Collaboration Meeting (DocDB 7829)



IPbus implementation

The structure of a waveform data packet

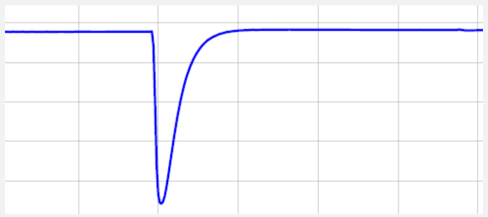
A waveform data packet is composed of a **header**, a **trailer** and the actual **waveform** data.
Each waveform contains **1 µs** of data sampled with **1GHz frequency**, for a total of **1000 samples**.

Data packet

Header

Fixed header start
Channel number (0,1,2)
Trigger window
Trigger count
Timestamp

Waveform data



1000 ADC samples

Trailer

Fixed trailer starting sequence
GCU ID
Fixed trailer end

1 waveform packet = 16 header bytes + 16 trailer bytes + 1 µs x 1000 ADC/µs x 2 bytes/ADC **= 2032 bytes**
The **FIFO** has a dimension of **8192 bytes** and can therefore contain **~4 waveforms**.

Maximum transferrable bandwidth for 1 GCU

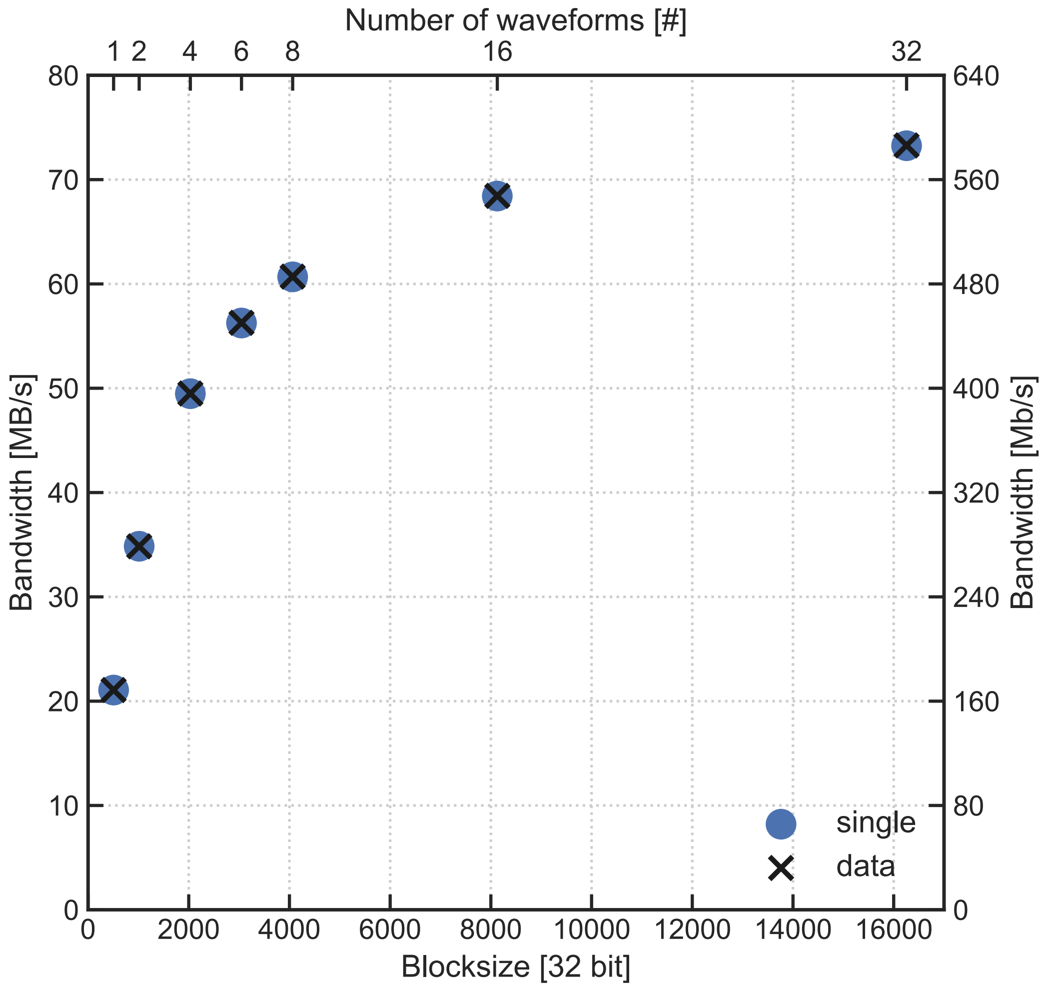
Via IPbus it is possible to **specify** the "**blocksize**" with which read the waveform data from the GCU.

Higher blocksize → less data packets → higher bandwidth

We performed a scan of the transferrable bandwidth as a function of the blocksize.

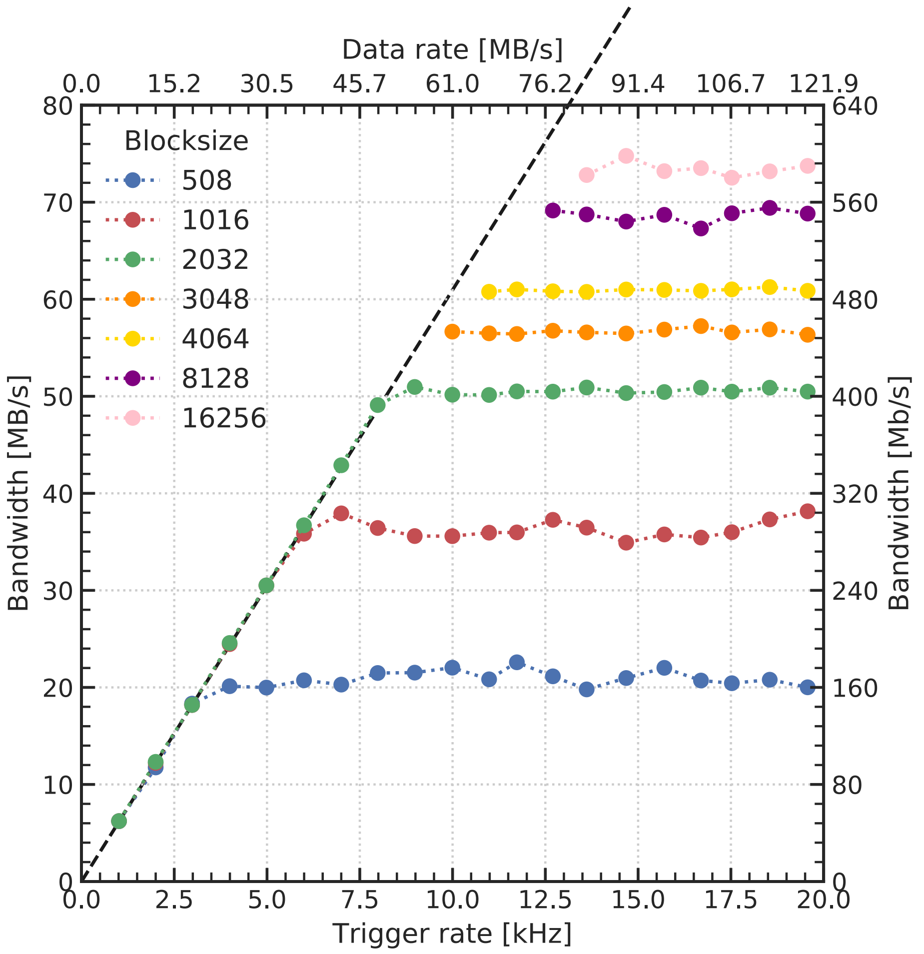
We logged both the network bandwidth (single) and the bandwidth used by valid waveform data (data).

- For large payloads it is possible to reach **~0.5 Gbit/s**
  → maximum permitted by IPbus protocol
- "single" bandwidth and "data" bandwidth correspond
  → all transferred packets are waveform data packets

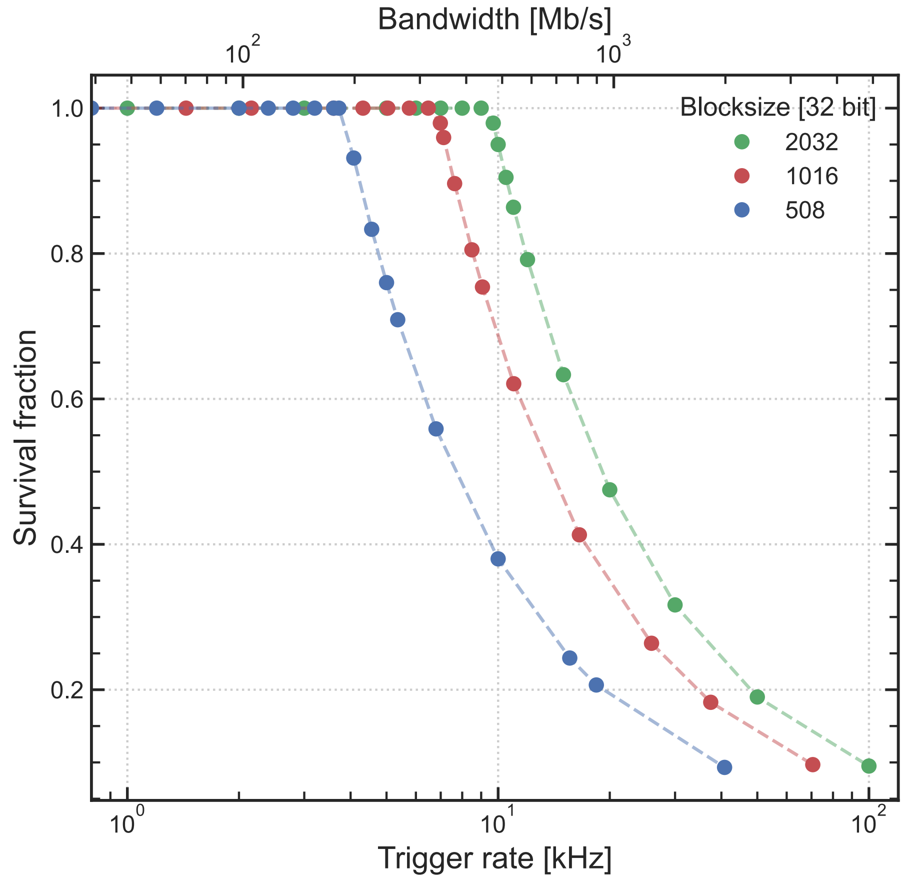


Maximum transferrable bandwidth for 1 GCU

Blocksize determines the maximum transferrable bandwidth for a GCU.
When **bandwidth saturates**, we are **not able to transfer all waveforms** (wf) collected by the GCU.



$$\text{survival fr.} = \frac{\text{\# wf transferred}}{\text{\# wf generated}}$$

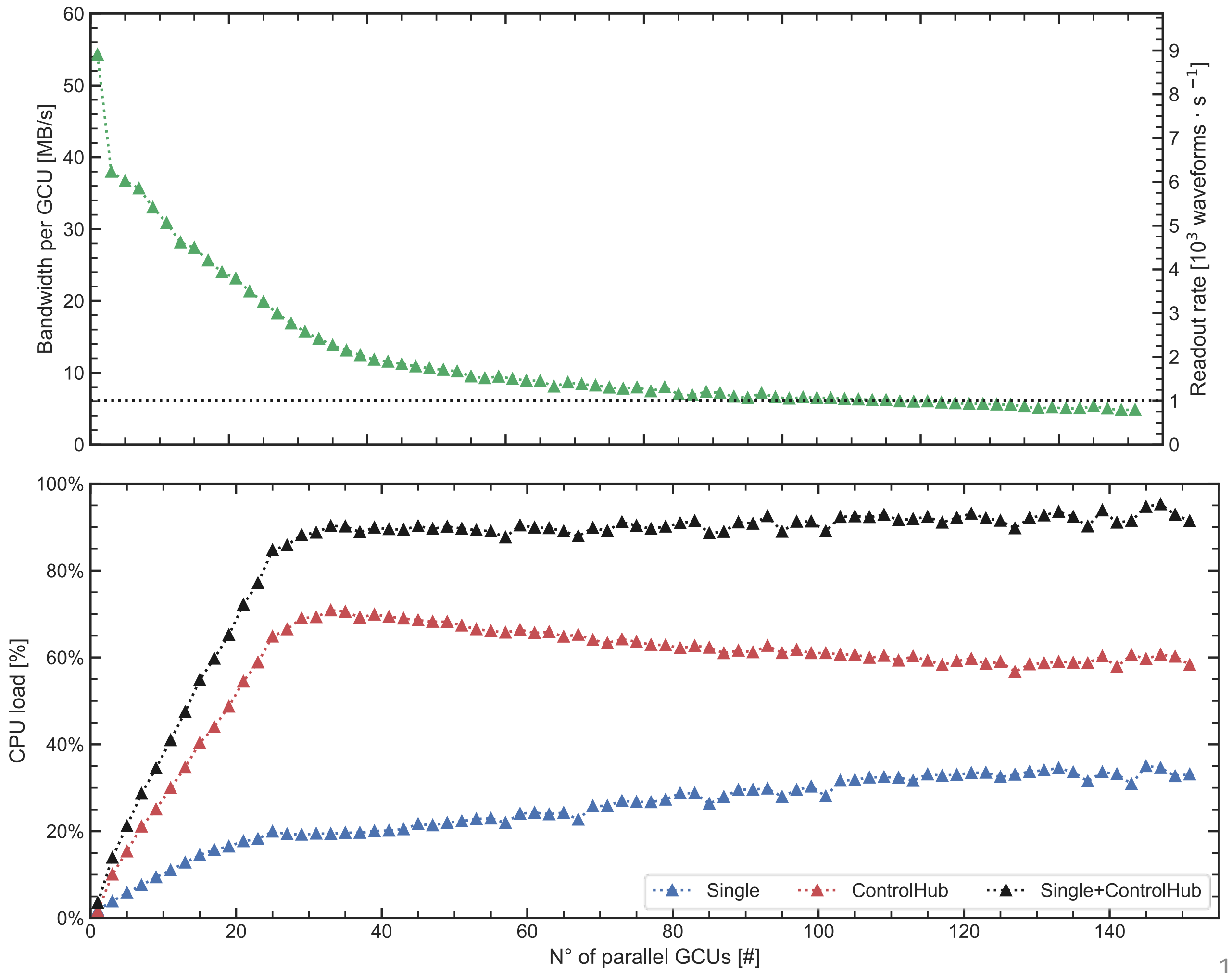


Managing several GCUs in parallel

At **Kunshan** we have a CentOS server having an Intel Xeon Gold 6226 CPU @ 2.70GHz for a total of **24 cores** (48 threads). We can test up to 344 GCUs in parallel.

We logged the transferred bandwidth as a function of the number of GCUs read in parallel for a blocksize of 2048.

- Server resources saturate for ~30 GCUs. Afterward, transferred bandwidth slowly continue to grow.
- Most resources are used by the ControlHub.
- Looking at the bandwidth transferred per GCU (total bandwidth/ # GCUs), it is still possible to manage a 1kHz trigger rate with 90-100 GCUs.

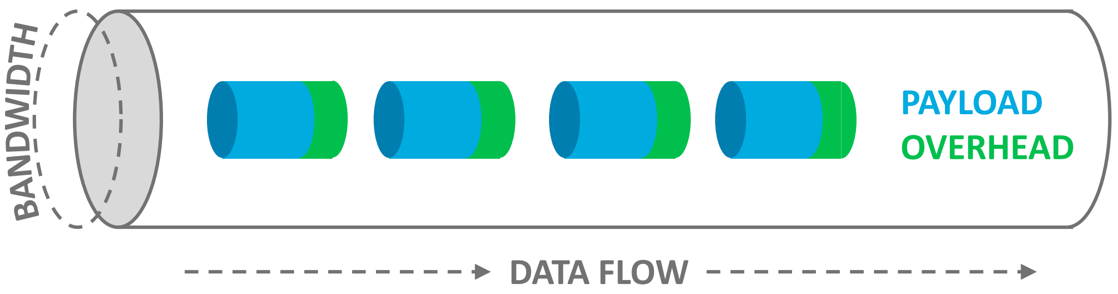


Reducing CPU usage with Jumbo Frames

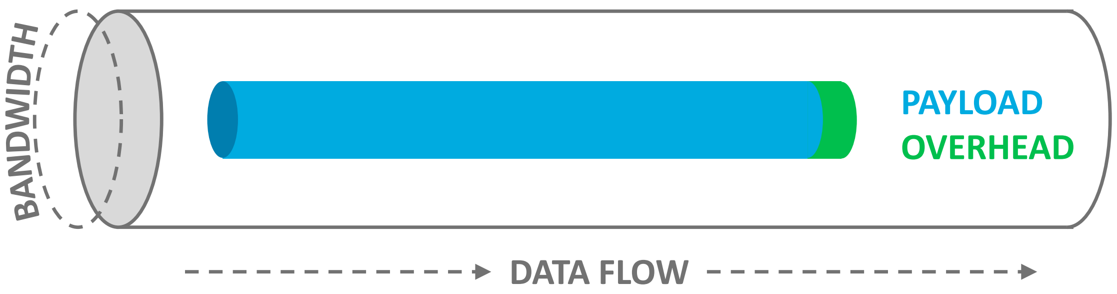
Standard ethernet packets are designed to carry **1500 bytes of payload**. This payload is also called **Maximum Transferrable Unit** (MTU).

Most modern network interfaces can support **MTUs up to 9000 bytes**. Suck ethernet packets are usually referred to as **Jumbo Frames.**

STANDARD FRAME (1500 MTU)



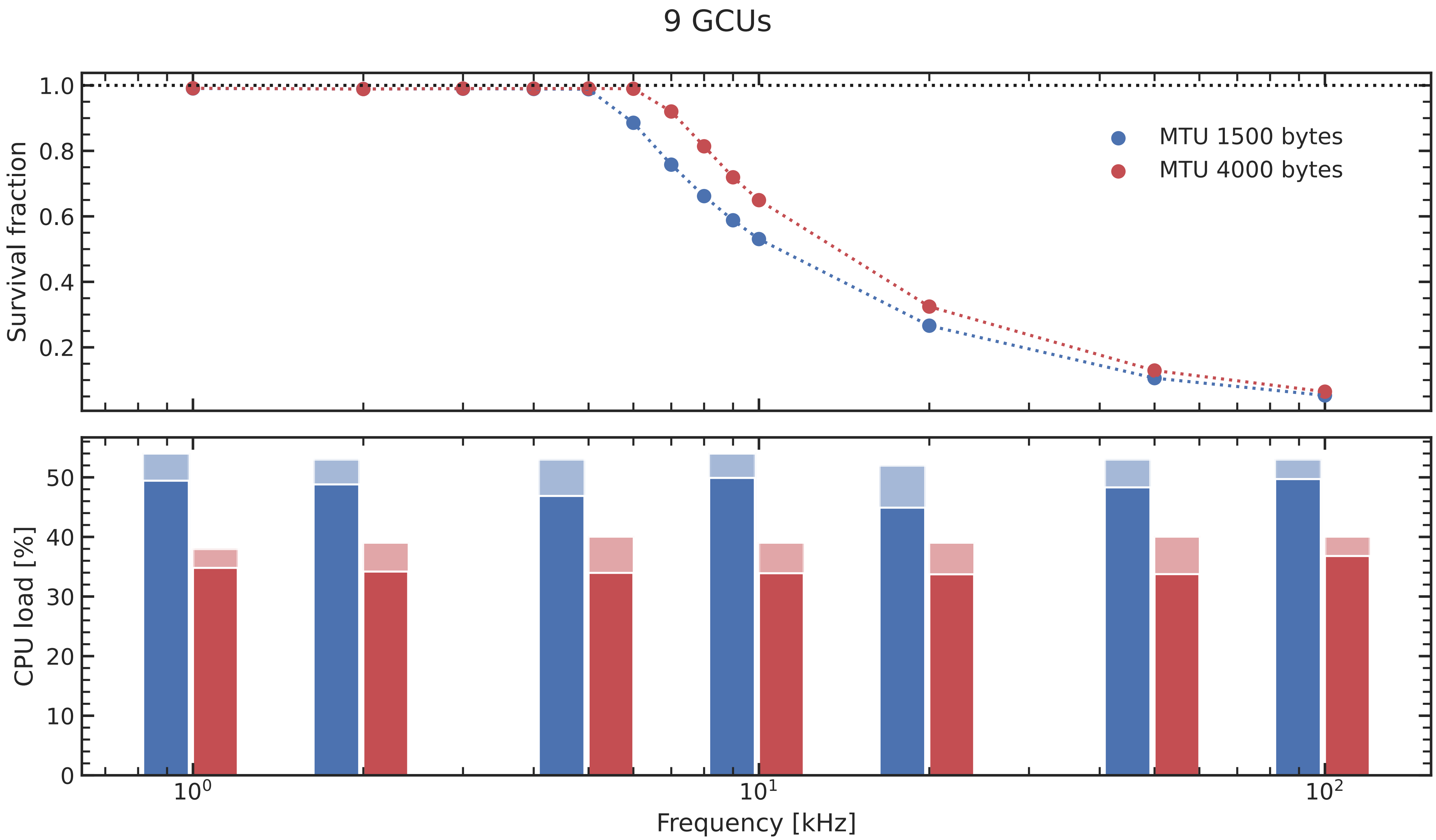
JUMBO FRAME (9000 MTU)



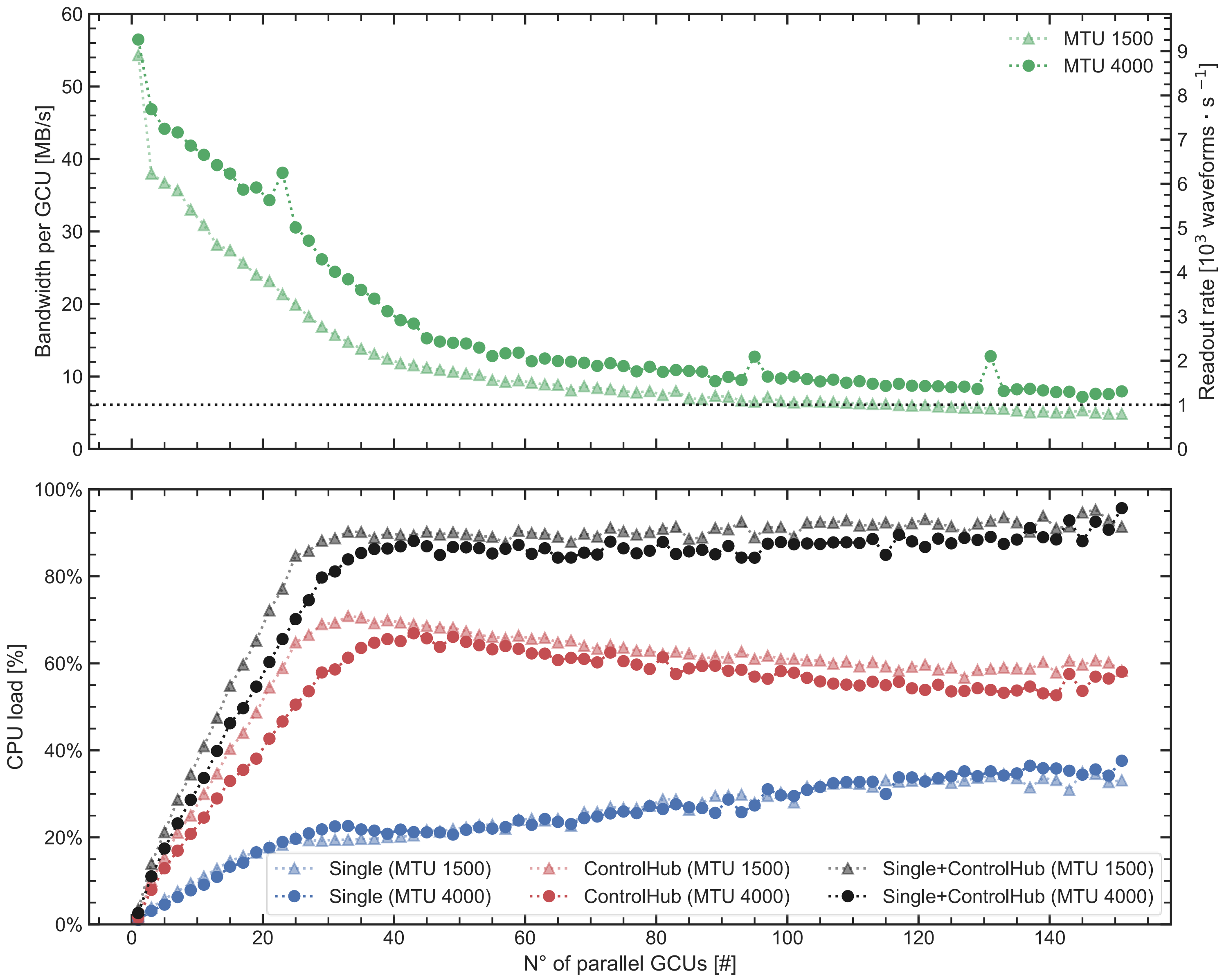
The rationale:

higher payloads → less packets → lower CPU utilization

Testing Jumbo Frames @ LNL



Testing Jumbo Frames @ Kunshan



Jumbo Frames permit to lower ControlHub's CPU utilization by 20%

As a consequence, it is now possible to manage a 1kHz trigger rate with 150 GCUs!

A rough estimate for JUNO setup:

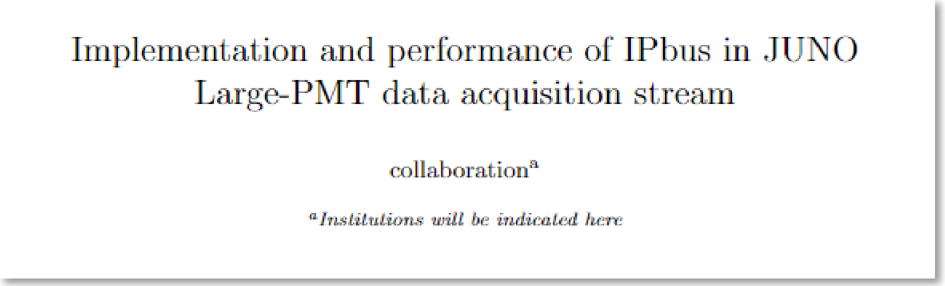
17612 L-PMTs
→ 17612/3 = 5871 GCUs
→ 5871/150 = 40 servers

Needs to be carefully checked!
Needs additional measurements!!

Final remarks

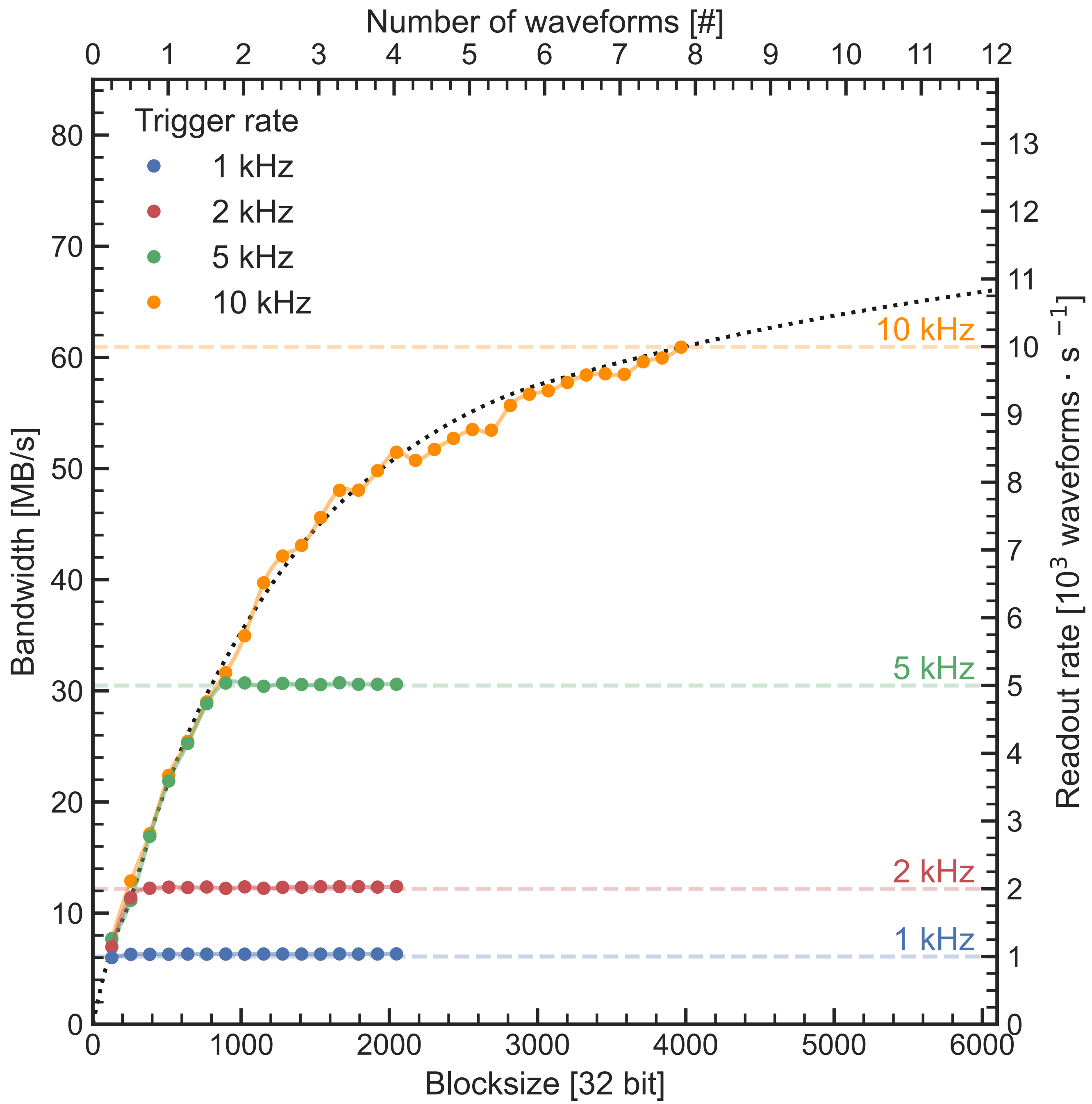
- At GCU level we can manage trigger rates up to 10 kHz.
- When dealing with multiple GCUs, the bottleneck is represented by the CPU usage of the ControlHub. However, keep in mind that ControlHub is crucial to permit the simultaneous access to data and slow control parameters.
- Jumbo Frames can help in reducing ControlHub's CPU usage. It is possible to sustain a trigger rate of 1kHz when managing 150 GCUs in parallel.

We are condensing all our tests and measurements in an article under preparation!


Implementation and performance of IPbus in JUNO
Large-PMT data acquisition stream

collaboration[a]

[a] *Institutions will be indicated here*



Number of waveforms [#]
0
1
2
3
4
5
6
7
8
9
10
11
12
Trigger rate
1 kHz
2 kHz
5 kHz
10 kHz
10 kHz
5 kHz
2 kHz
1 kHz
Bandwidth [MB/s]
0
10
20
30
40
50
60
70
80
Readout rate [10^3 waveforms · s^{-1}]
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Blocksize [32 bit]
0
1000
2000
3000
4000
5000
6000