Performance profiling and design choices of an RDMA implementation using FPGA devices

TWEPP 2023

Oct 1 – 6, 2023
Geremeas, Sardinia, Italy

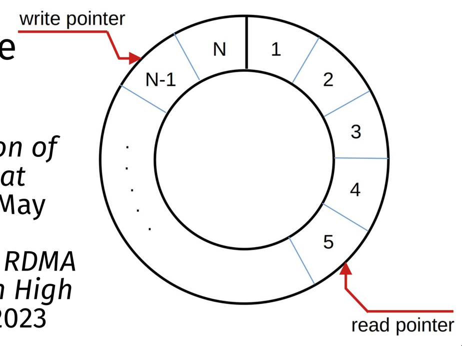
Matei-Eugen Vasile[1], Voichita Iancu[2], Sorin Martoiu[1], Gabriel Stoicea[1], Nayib Boukadida[3], Radu Hobincu[2]

[1]IFIN-HH, [2]UPB, [3]Nikhef

1. Design choices

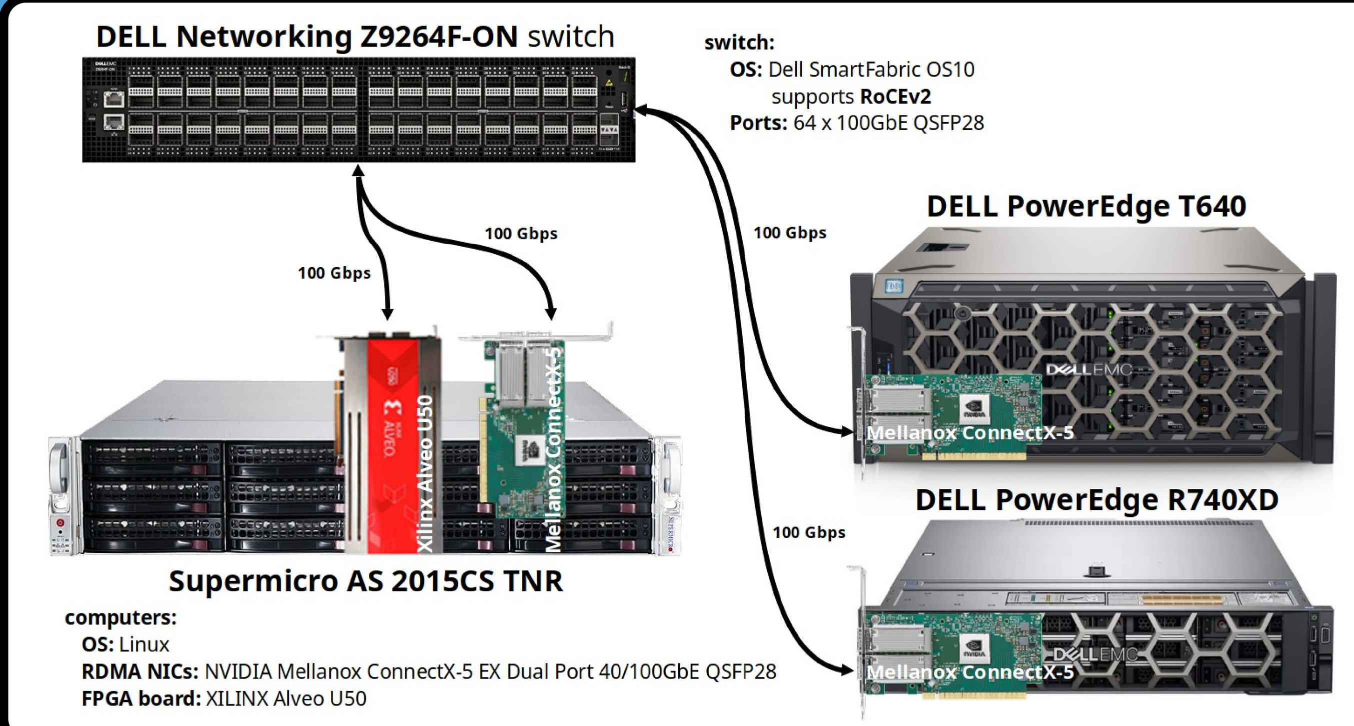
- **RDMA** stands for **Remote Direct Memory Access** - used for data transfers with as little CPU involvement as possible
  - several *transport services*, such as *reliable connection*, *unreliable connection* etc. The one used here is **reliable connection**, which it is **similar to TCP/IP**;
  - several *transport functions*, such as *send/receive*, *write* and *read*. The one used here is **RDMA write**;
- up to this point[1,2], tested for **individual bursts** of data
- in a **production setting**, that is not enough – for **meaningful** test results, several other issues need to be taken into account:
  - **multiple simultaneous** clients and connections
  - **continuous flow** of data
  - accounting for **consumption** of data by clients
- **multiple simultaneous** clients and connections
  - **senders** implemented **both** in **software** and in **hardware** on **Xilinx Alveo** boards for **individual bursts**
- **continuous flow** of data can't be safely implemented when using **RDMA write** without accounting for **consumption** of received data by clients
- to accomplish it, implemented **solution** using:
  - **circular buffer**
  - **flow control**
- currently only implemented for **software senders**
- there are **only two receivers**, one for **individual bursts**, and one for **continuous flow**, implemented in software, working for **all types of senders**

- **backpressure** mechanism:
  - **stops transfer** when **circular buffer occupancy** goes over **threshold** and **restarts** it when **occupancy goes back down**
  - **upper threshold** and **lower threshold** pair to avoid flip-flopping
  - **out-of-band sender/receiver** communication using **TCP/IP**
- **receiver – 2 threads**, syncronized using a semaphore:
  - **#1** receives data write notification, posts semaphore and activates backpressure
  - **#2** waits semaphore, reads data and deactivates backpressure
- **sender – 3 threads**, 2 of them syncronized using a semaphore:
  - **#1** sends data, posts semaphore
  - **#2** waits semaphore, sends data write notification
  - **#3** receives backpressure commands



1) M. Vasile et. al *FPGA implementation of RDMA for ATLAS readout with FELIX at high luminosity LHC*, JINST, vol. 17, May 2022
2) M. Vasile et. al *Integration of FPGA RDMA into the ATLAS readout with FELIX in High Luminosity LHC*, JINST, vol. 18, Jan 2023

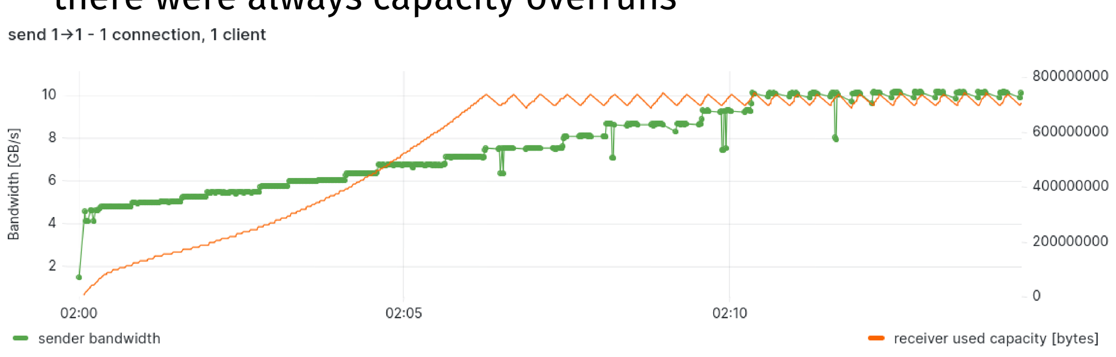
2. Development setup

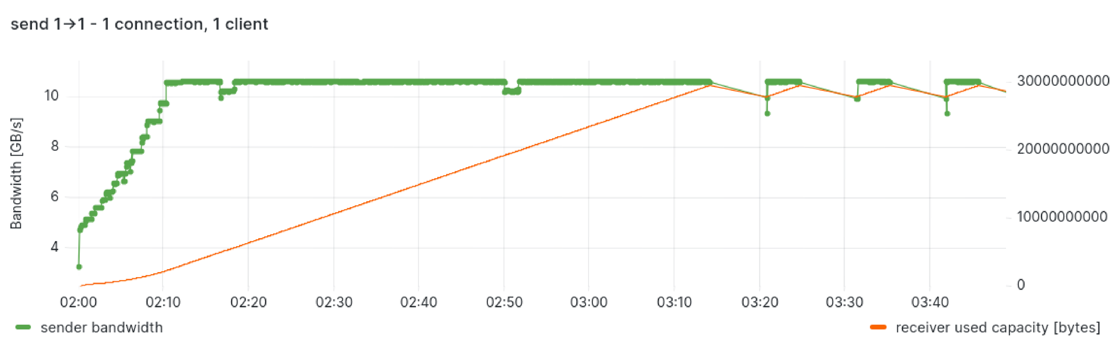


3. Design consequences

- **burst = message size {bytes] x message count**
- tested[3] with **8192x100, 8192x1000, 32768x100, 32768x1000** bursts
  - less than **8192b** or **100 msgs.** – no full bandwidth use
- tested with **circular buffer capacities**: **10, 100, 1000**
  - **10** - too small, no matter what other parameters were used, there were always capacity overruns

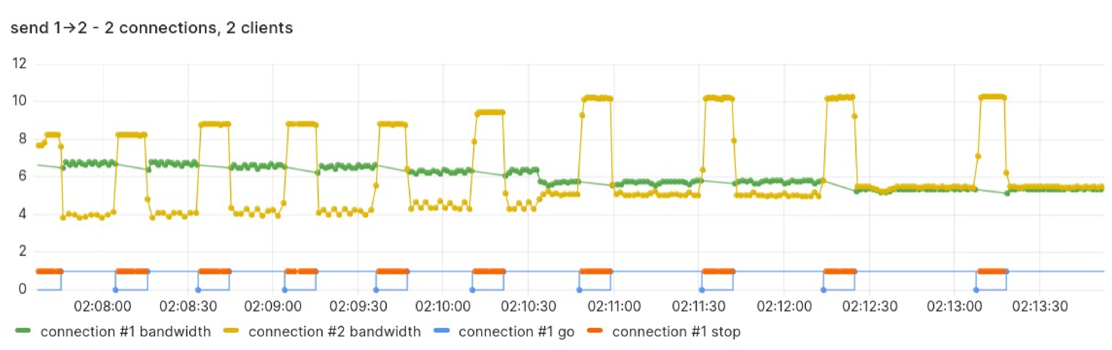
- once buffer occupancy reaches the **upper backpressure threshold for the first time**, the transfer mechanism settles in a **periodic pattern** between the upper and the lower thresholds
  - **only way to prevent it**: have a **client that can read faster than the sender can write**

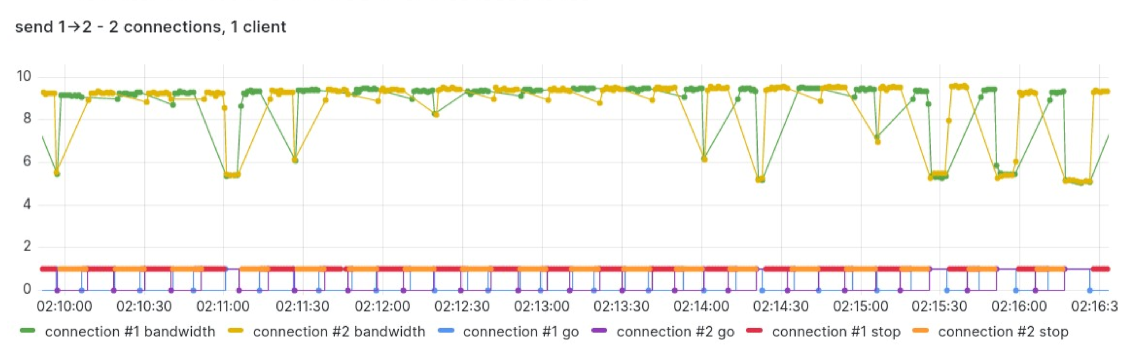




- when the **sender** is using **multiple connections** to **one or multiple clients**, the **available sender bandwidth** will be **split** across all connections
  - **all connections - same client**: bandwidth split **equally**
  - **different clients**: bandwidth can be split **unequally** between connections ending on different clients

- on the **PCs in the development setup**, if the sender **sends data** with **less than** about **5GB/s**, the receiver will be able to **read** it **fast enough** so that **backpressure** is **never triggered**
  - **backpressure not triggered → send and receive bandwidths** are almost **equal** (i.e. little overhead)
  - **backpressure triggered → send bandwidth** is roughly **double** the **receive bandwidth**



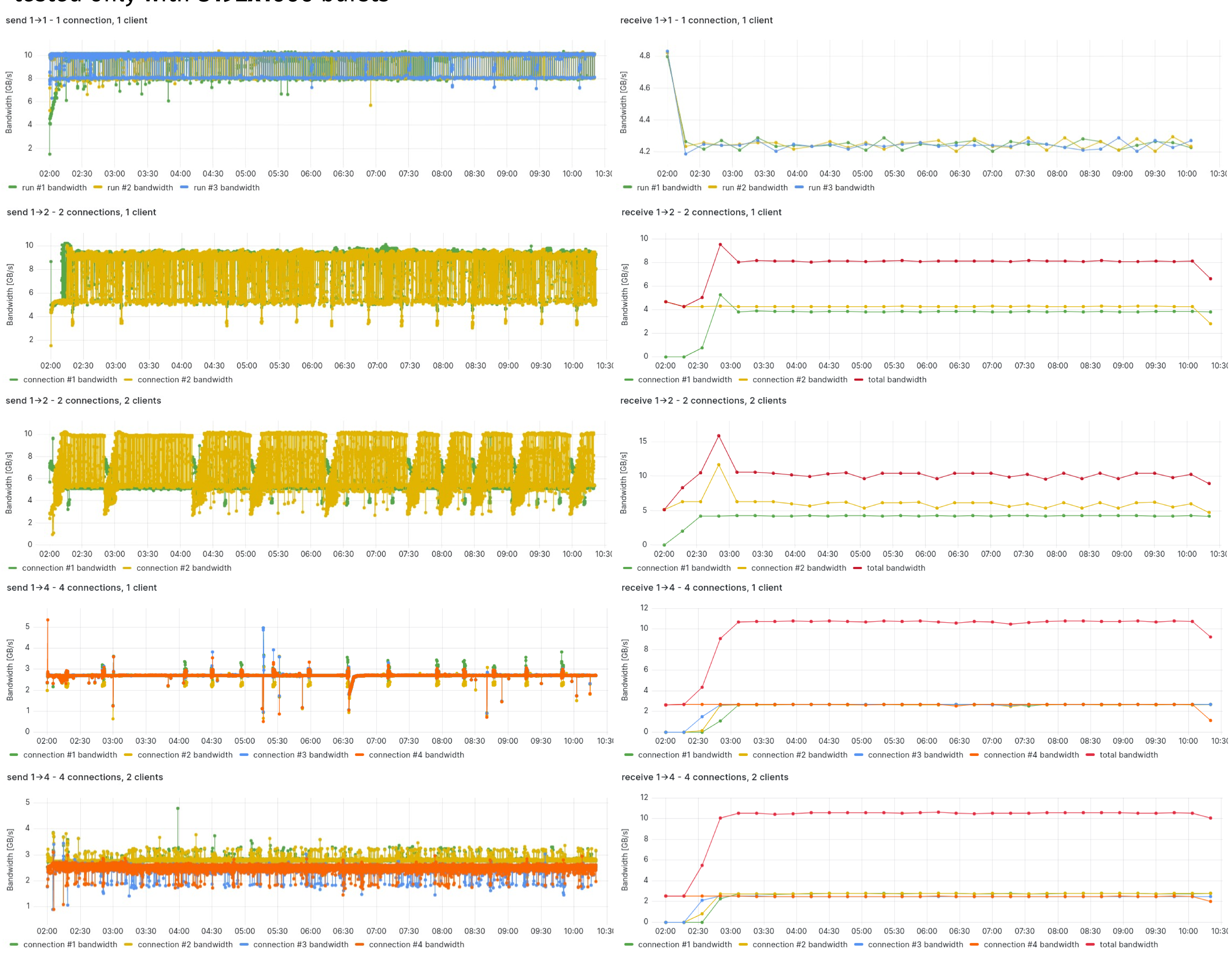


3) The code of the software senders and all the receivers can be found at https://github.com/mev/rdmatools

4. Continuous flow, multiple connections (software)

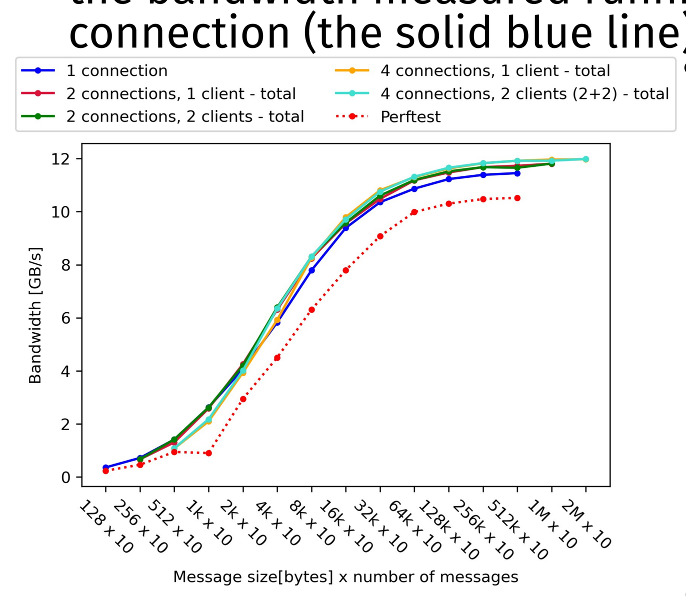
- all the tests run with **1 sender**
- **independent flow control** on **each connection**
- 1 connection → 1 client - tested with **8192x100, 8192x1000, 32768x100, 32768x1000** bursts
- 2 connections → client, 2 connections → 2 clients, 4 connections → 1 client, 4 connections → 2 clients – tested only with **8192x1000** bursts

- tested 2 threshold configurations:
  - **90/85** (90% buffer occupancy for upper threshold and 85% buffer occupancy for lower threshold)
  - **80/75**
- **no noticeable performance difference**
- the only **important parameter** seems to be the the **upper threshold** so that **no capacity overruns** happen

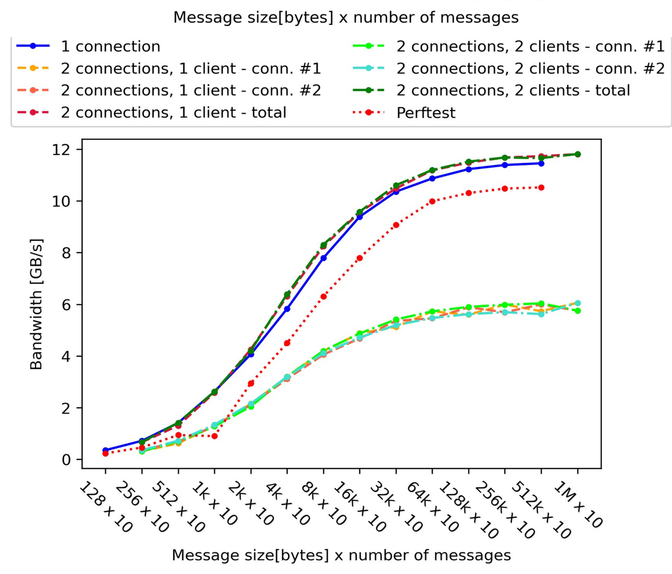


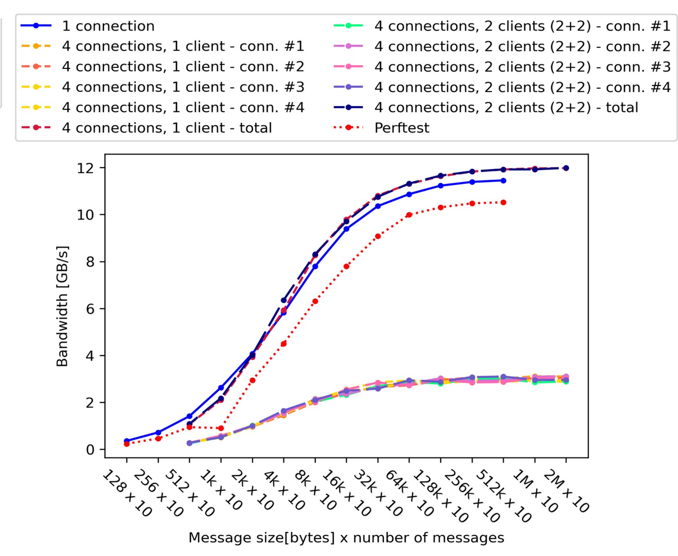
5. Multiple connections (hardware)

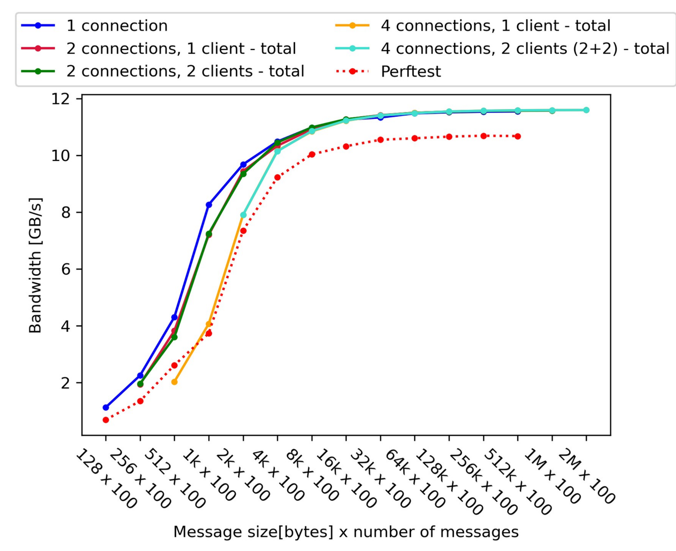
- the hardware implementation multiple connections feature was initially developed to run individual burst tests
  - as a consequence, the independent control of each connection was not a priority when this was developed
  - continuous flow requires fully independent control for each connection - this will be implemented in the future
- tested with **message sizes** (in *bits*): **128** to **512M**
- tested with **message counts**: **10, 100, 1000**
- two references used:
  - the bandwidth reported by the *ib_write_bw* test from the *Perftest* package (the red dotted line)
  - the bandwidth measured running the test with a single connection (the solid blue line)



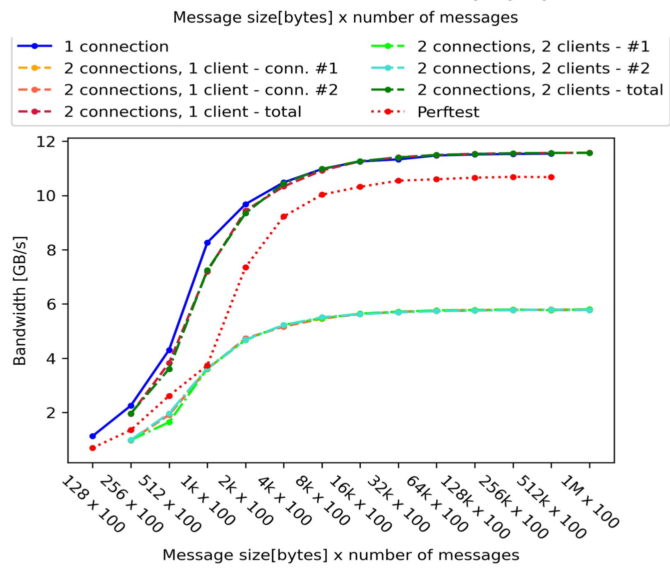
- **10** msg. count, tested with:
  - two connections distributed as:
    - both conn. on the same client
    - two clients, each with a single conn.
  - four connections distributed as:
    - all four conn. on the same client
    - two clients, each with two conn.
  - at **link saturation**, the total send bandwidth is **higher** than the single connection setup by:
    - **2.41% for 2 connections, 1 client**
    - **1.76% for 2 connections, 2 clients**
    - **4.05% for 4 connections, 1 client**
    - **4.06% for 4 connections, 2 clients**

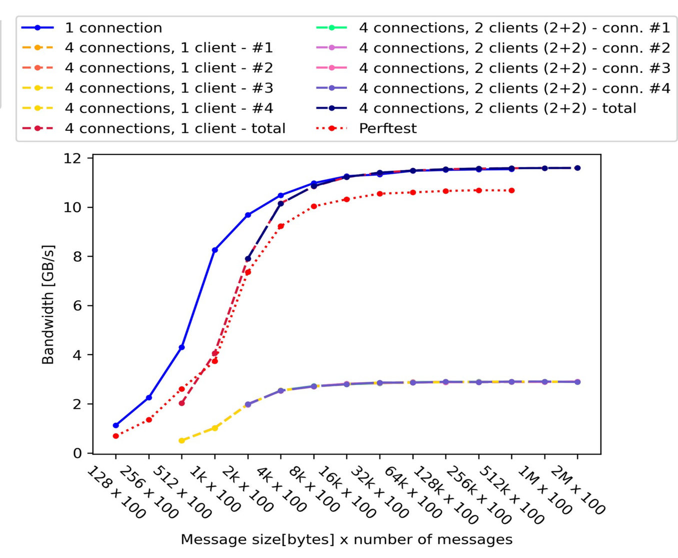


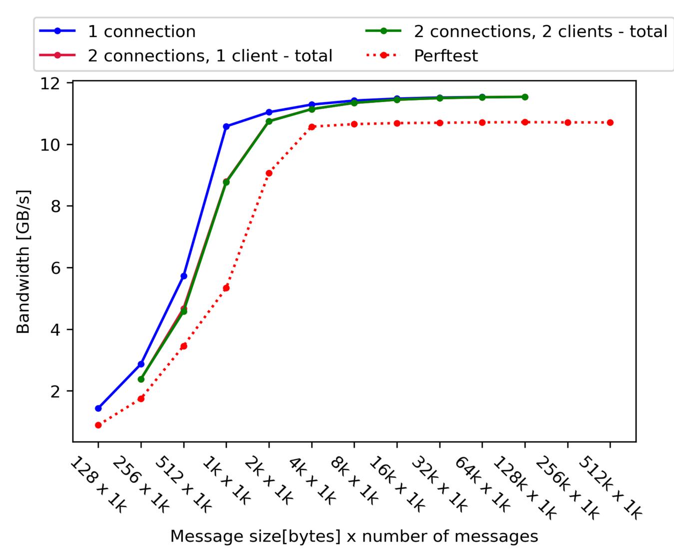




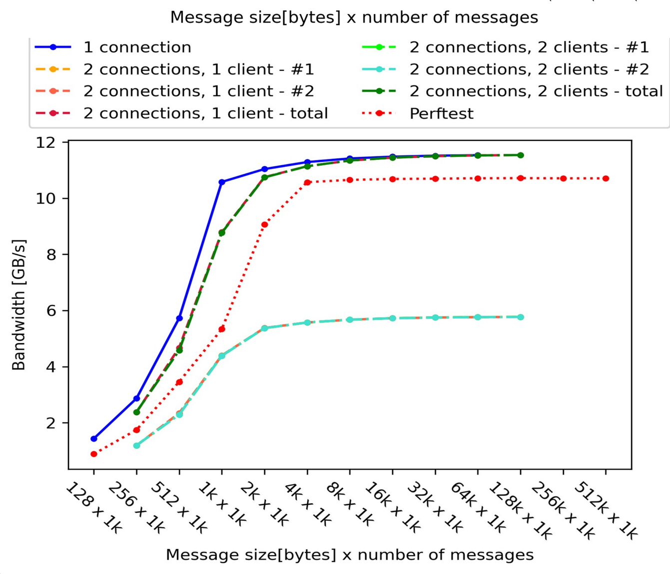
- **100** msg. count, tested with:
  - two connections distributed as:
    - both conn. on the same client
    - two clients, each with a single conn.
  - four connections distributed as:
    - all four conn. on the same client
    - two clients, each with two conn.
  - at **link saturation**, the total send bandwidth is higher than the single connection setup by:
    - **0.23% for 2 connections, 1 client**
    - **0.22% for 2 connections, 2 clients**
    - **0.34% for 4 connections, 1 client**
    - **0.34% for 4 connections, 2 clients**







- **1000** msg. count, tested with:
  - two connections distributed as:
    - both conn. on the same client
    - two clients, each with a single conn.
  - all four connections test setups are currently overloading the resources of the FPGA RDMA core implementation
  - at **link saturation**, the total send bandwidth is **lower** than the single connection setup by:
    - **0.31% for 2 connections, 1 client**
    - **0.31% for 2 connections, 2 clients**



- the theoretical maximum bandwidth of the used links is **100Gb/s** (i.e. **12.5GB/s**)
- a software implementation, both ours and what can be measured with *Perftest*, can reach up to **10.5GB/s**
- our hardware implementation has been measured to reach up to **11.54GB/s** with a single connection and up to **11.98GB/s** total with multiple connections