CUG 2021

VkFFT: Performant, Cross-Platform and Open-Source GPU FFT Library

11.05.2021 I Dmitrii Tolmachev I ETH Zürich

dtolm96@gmail.com I dmitrii.tolmachev@erdw.ethz.ch

There are many Fast Fourier Transform libraries, what is so special about VkFFT?

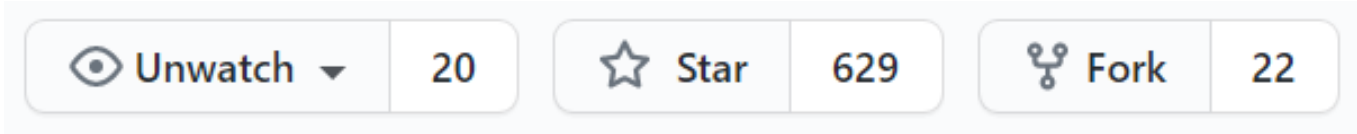
1. VkFFT is architecture, OS and API cross-platform
2. VkFFT is extremely performant
3. VkFFT is released in open-source
4. VkFFT supports many novel optimization techniques

VkFFT: current features

- 1D/2D/3D systems
- Maximum dimension size is 2^32
- Radix-2/3/5/7/11/13 Stockham autosort algorithm
- In-place and out-of-place transformations
- Batch execution
- Complex to complex, real to complex and complex to real transformations
- Convolutions
- Native zero padding
- Single/double/half precision support

VkFFT is available on GitHub:

https://github.com/DTolm/VkFFT



VkFFT is architecture cross-platform:

- Raspberry Pi 4 GPU
- Mobile GPUs: Adreno, Mali, PowerVR
- Integrated GPUs: Intel UHD, AMD Vega
- Dedicated GPUs: Nvidia (from GeForce 600), AMD (from Radeon HD 7000)
- HPC GPUs: Nvidia A100 and AMD MI100

VkFFT is OS cross-platform:

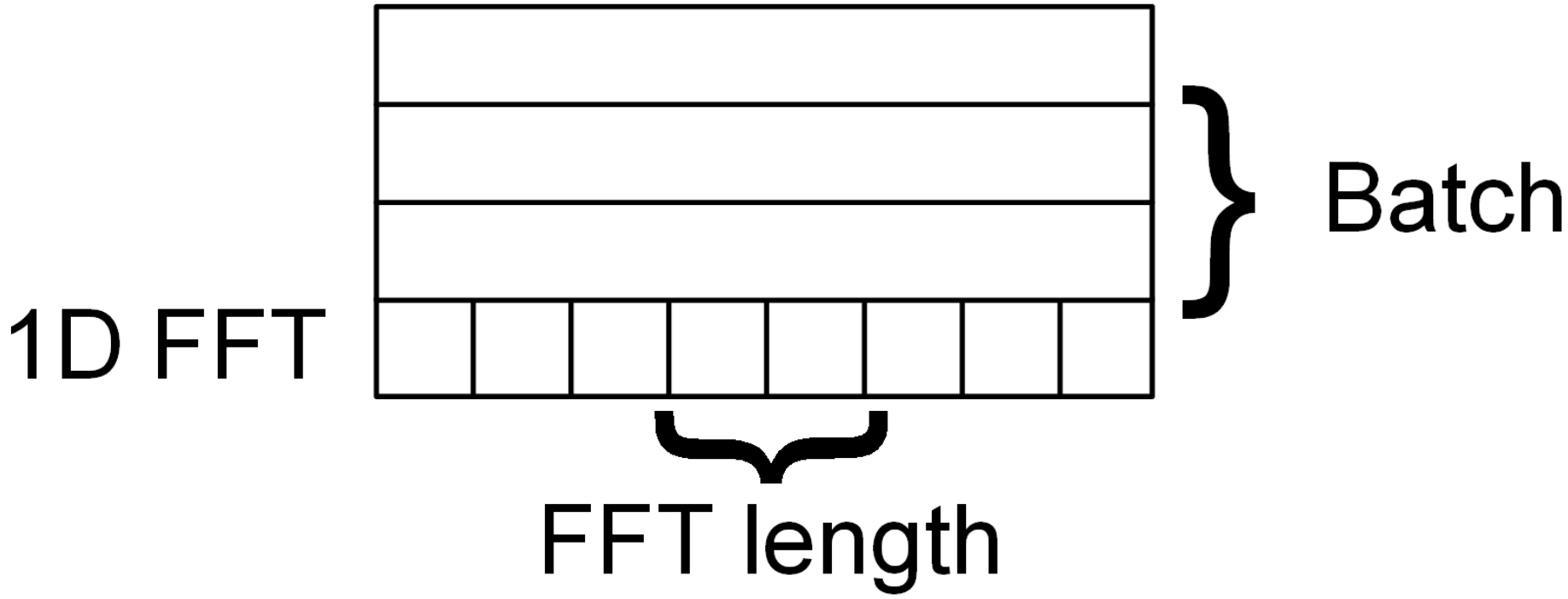
- Linux
- Windows
- Android
- macOS and iOS (through a compatibility layer)

VkFFT is API cross-platform:

- Vulkan
- CUDA
- HIP
- OpenCL
- More (DirectX, OpenGL) to come!

Performance tests configurations:

1. Batched 1D FP32 FFT on the range from 2 to 4096
2. Batched 1D FP64 FFT on the range from 2 to 4096



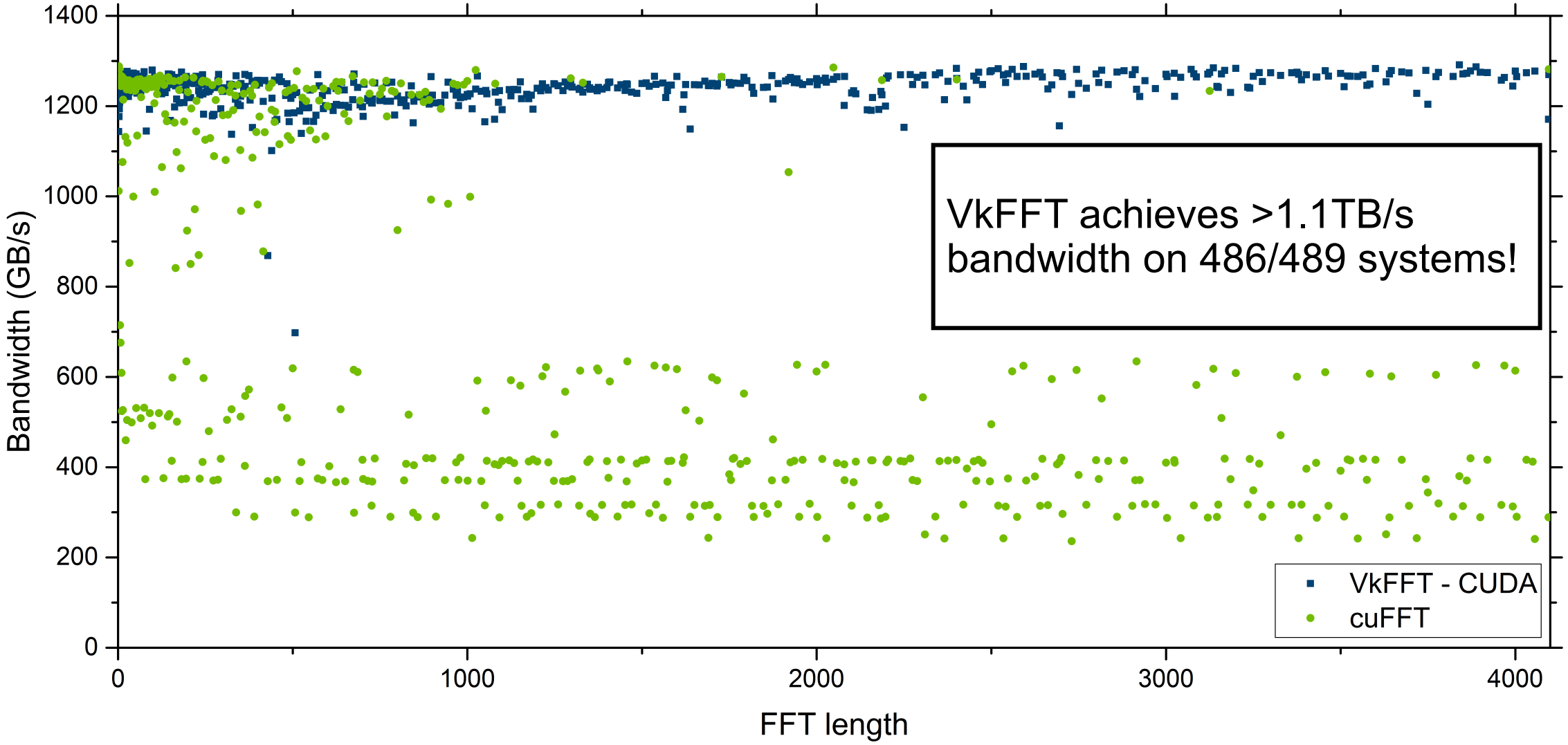
Performance tests configurations:

1. Batched 1D FP32 FFT on the range from 2 to 4096
2. Batched 1D FP64 FFT on the range from 2 to 4096
3. 3D cube FP32 FFT on the range from 2^3 to 512^3

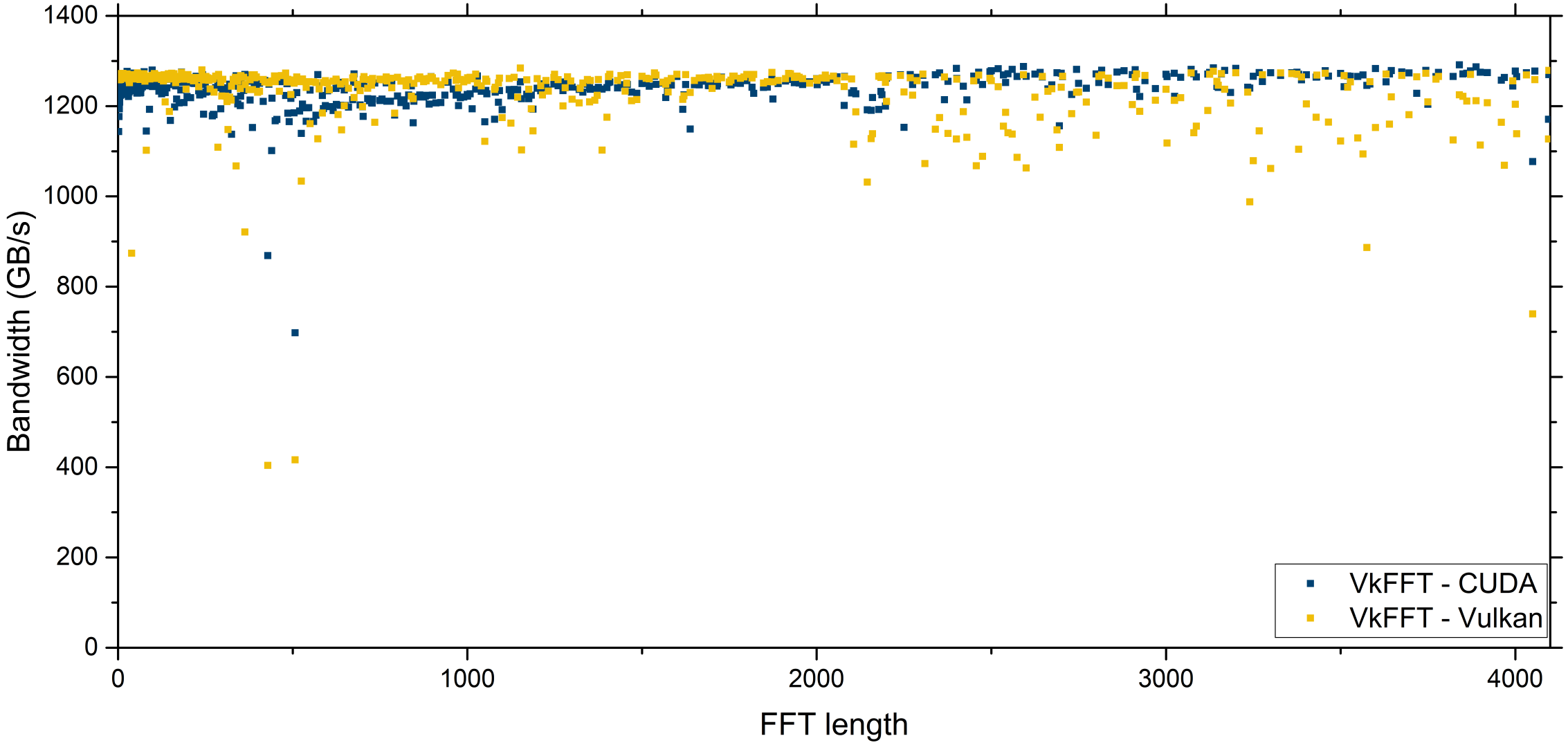
Tested GPUs: Nvidia A100 and AMD MI100

Tested APIs: Vulkan, CUDA and HIP

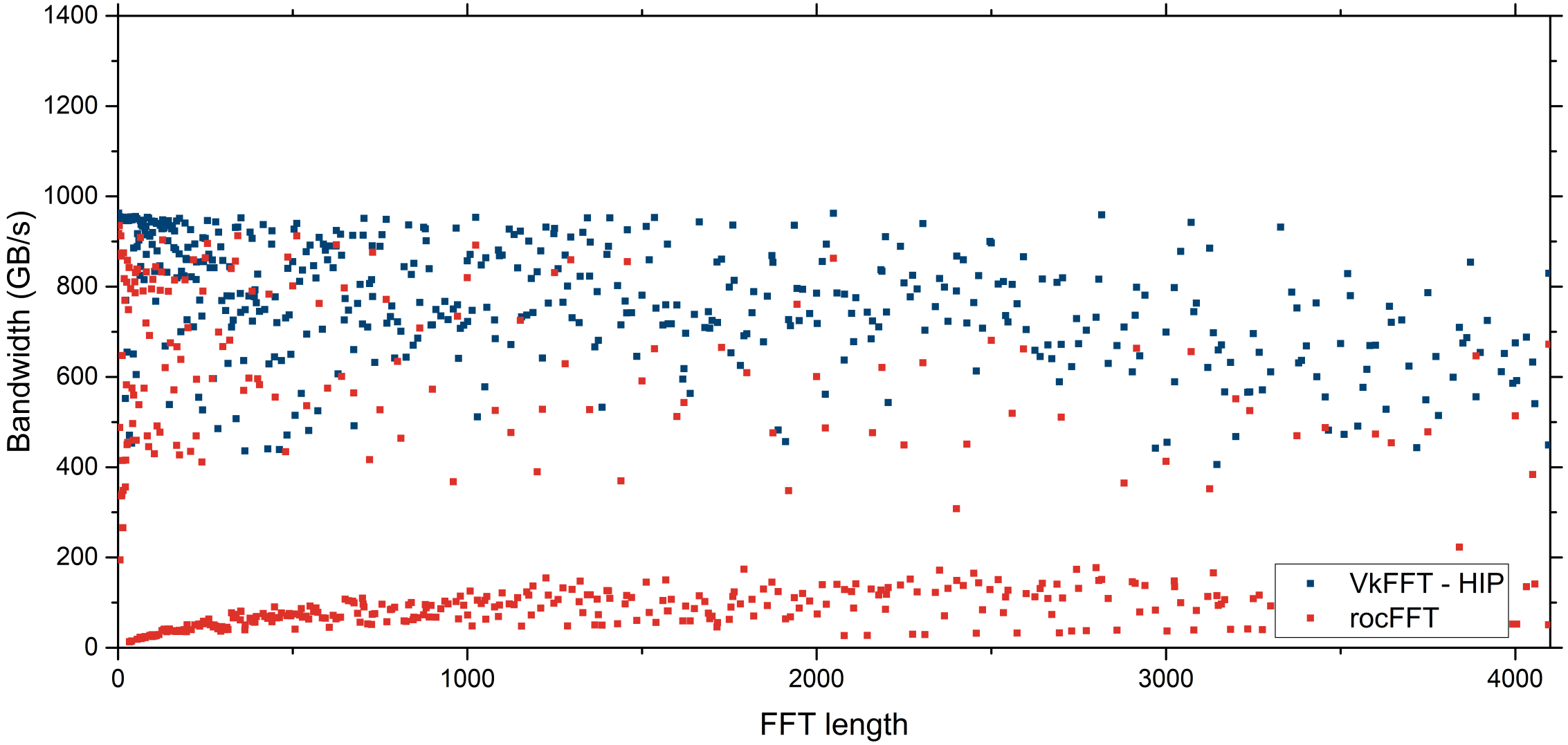
Batched 1D FP32 FFT on Nvidia A100



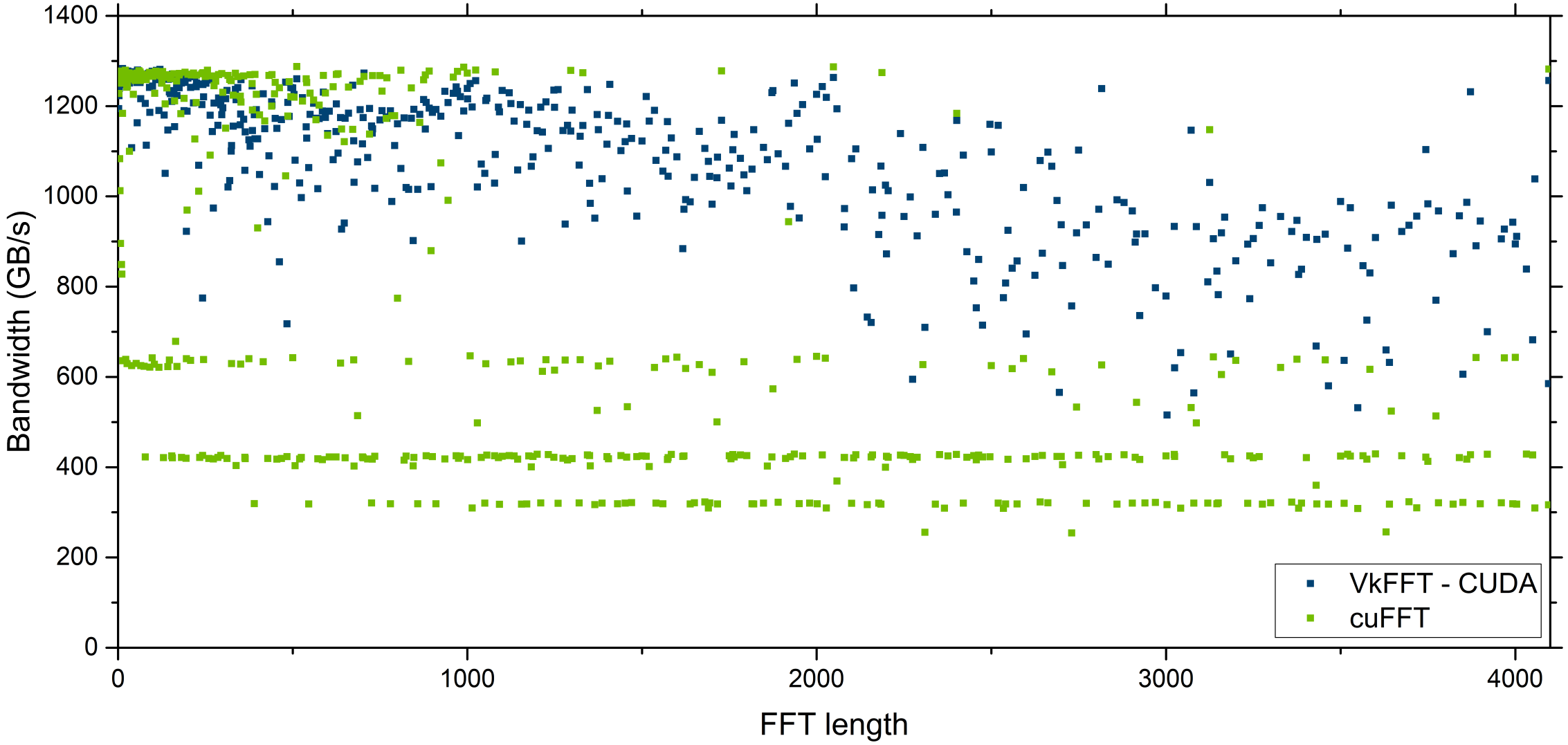
Batched 1D FP32 FFT on Nvidia A100



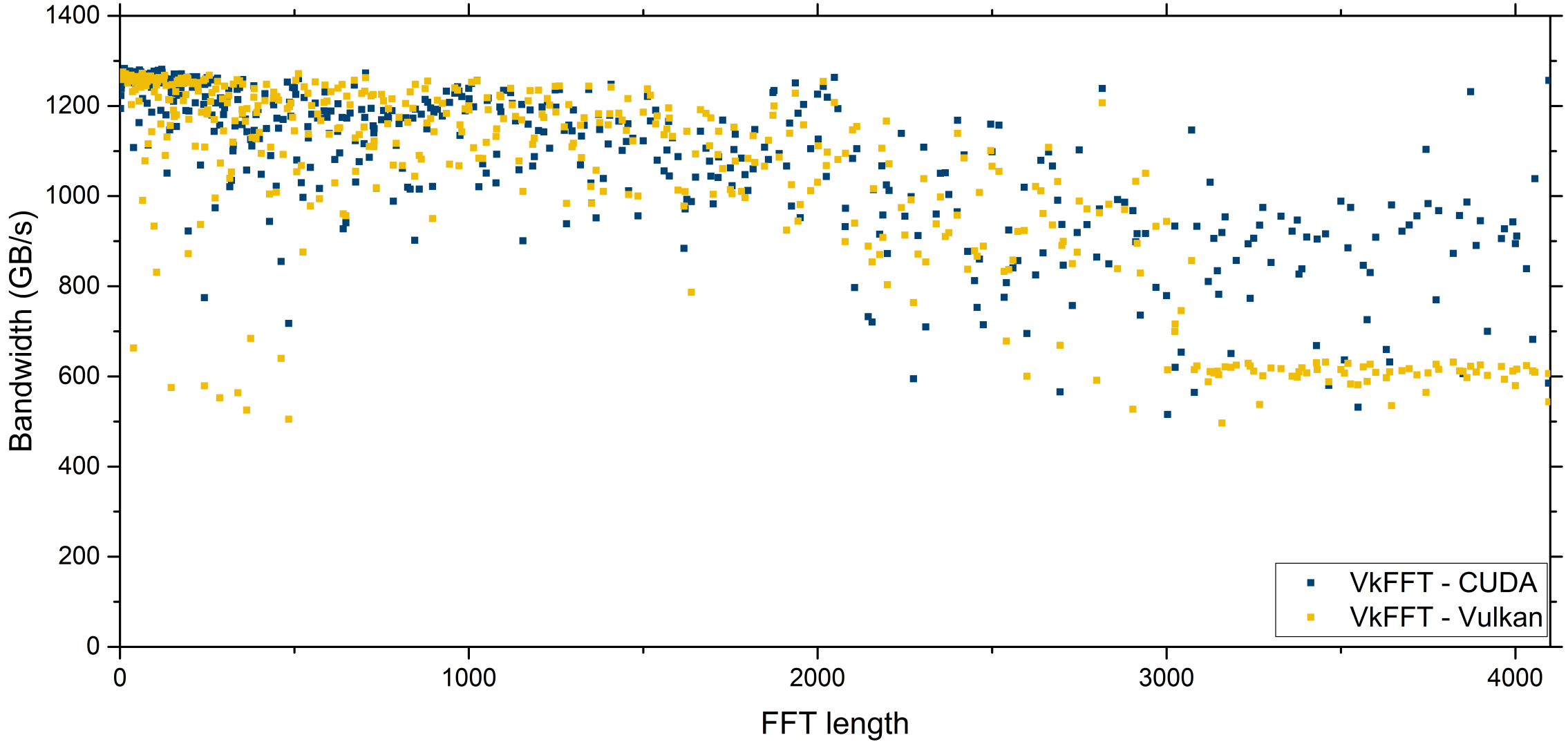
Batched 1D FP32 FFT on AMD MI100



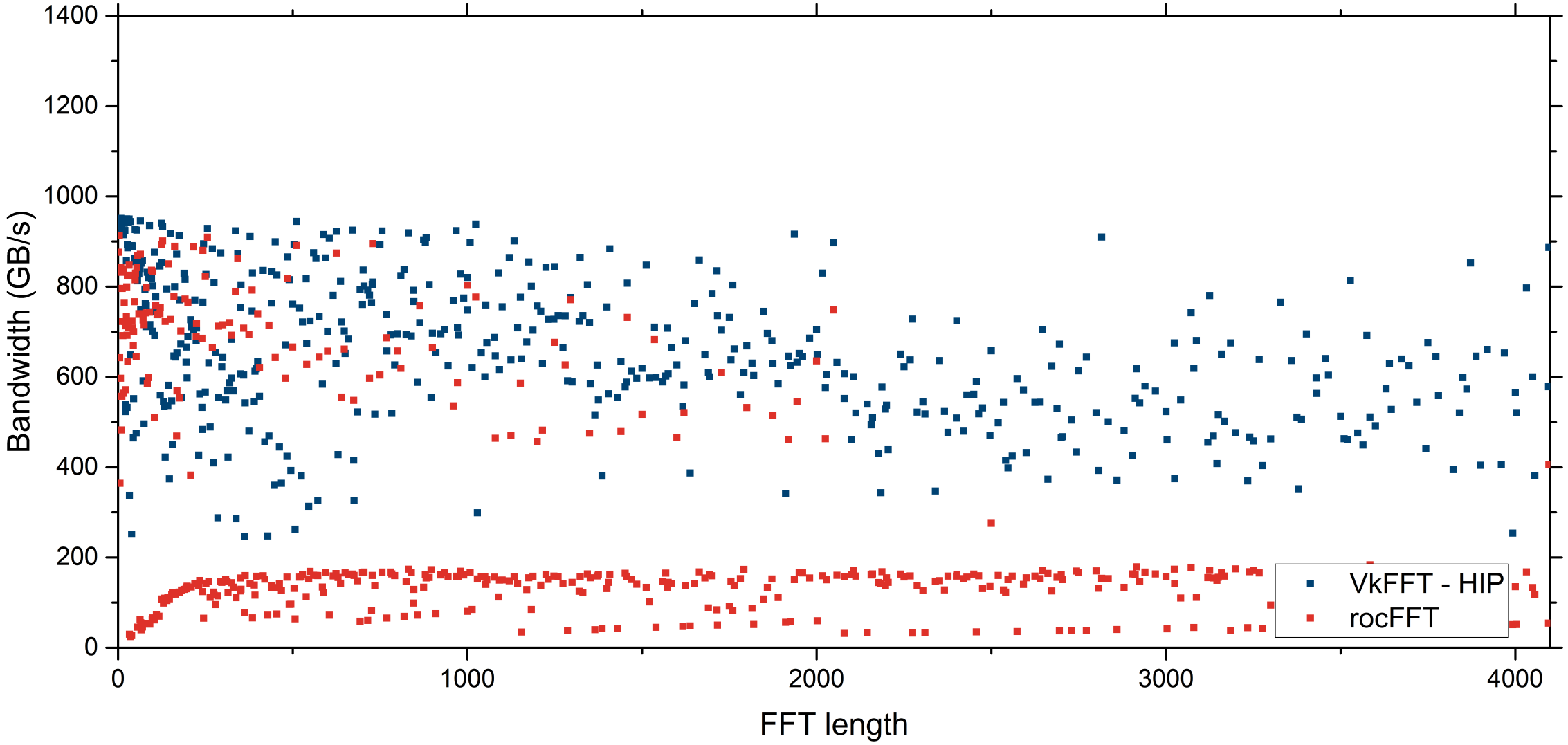
Batched 1D FP64 FFT on Nvidia A100



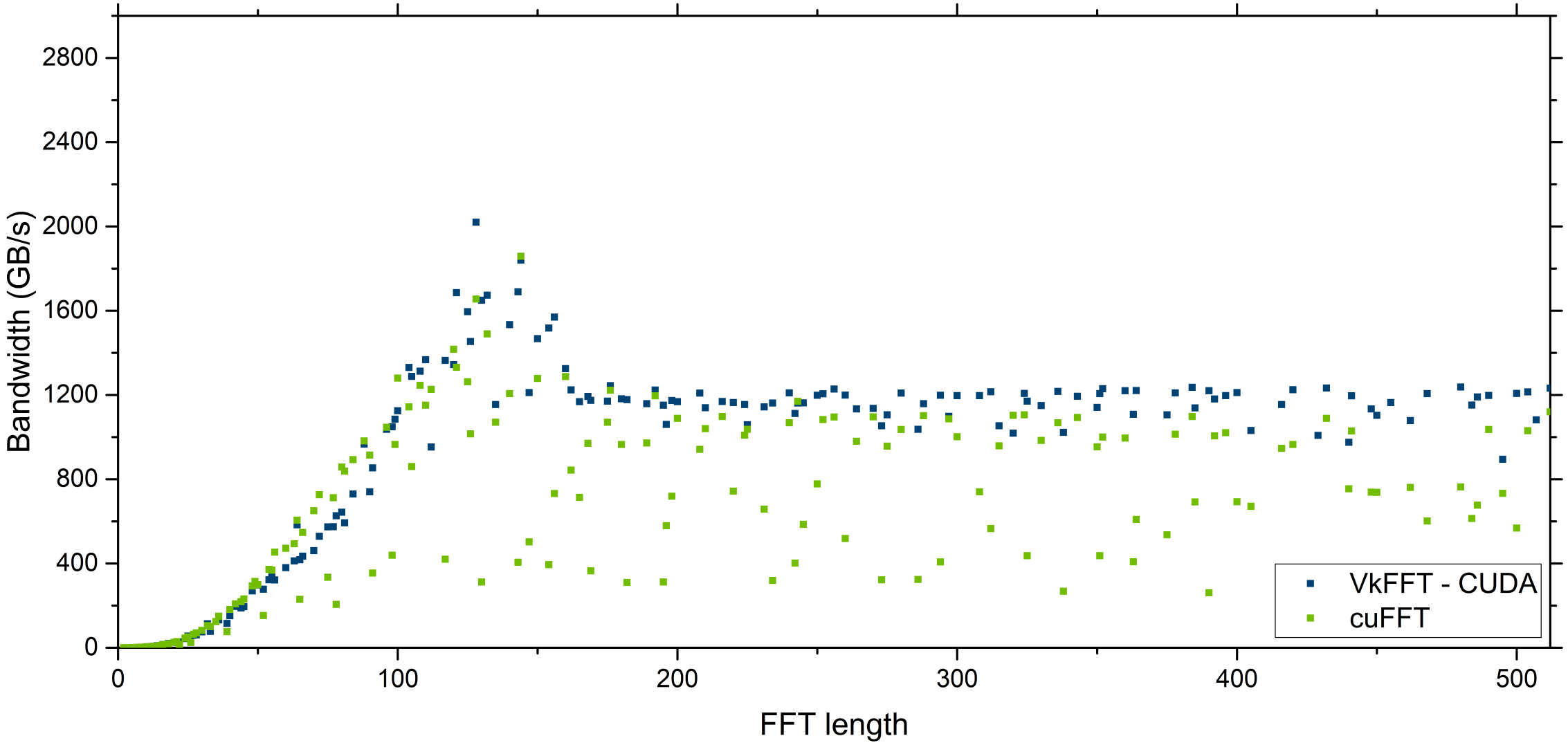
Batched 1D FP64 FFT on Nvidia A100



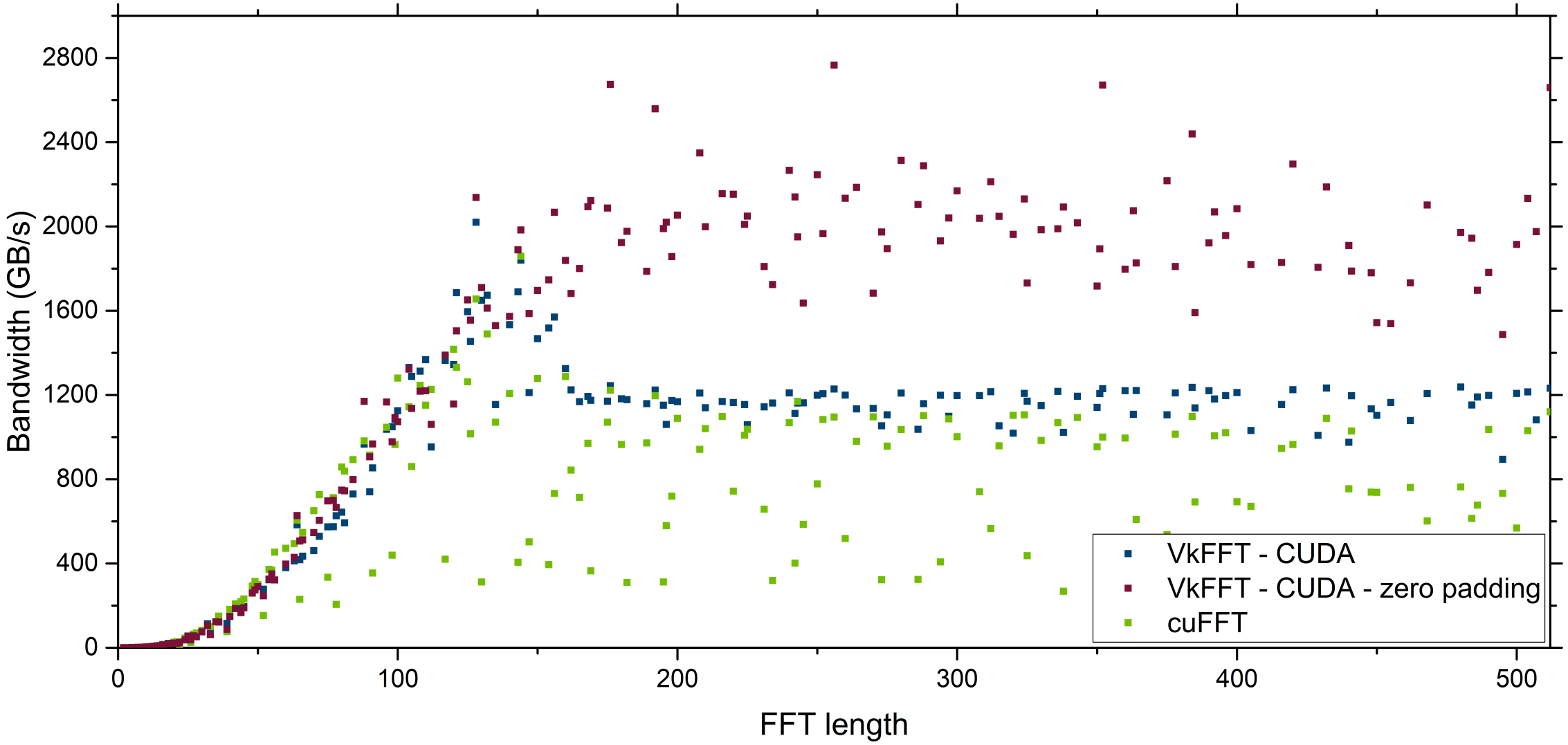
Batched 1D FP64 FFT on AMD MI100



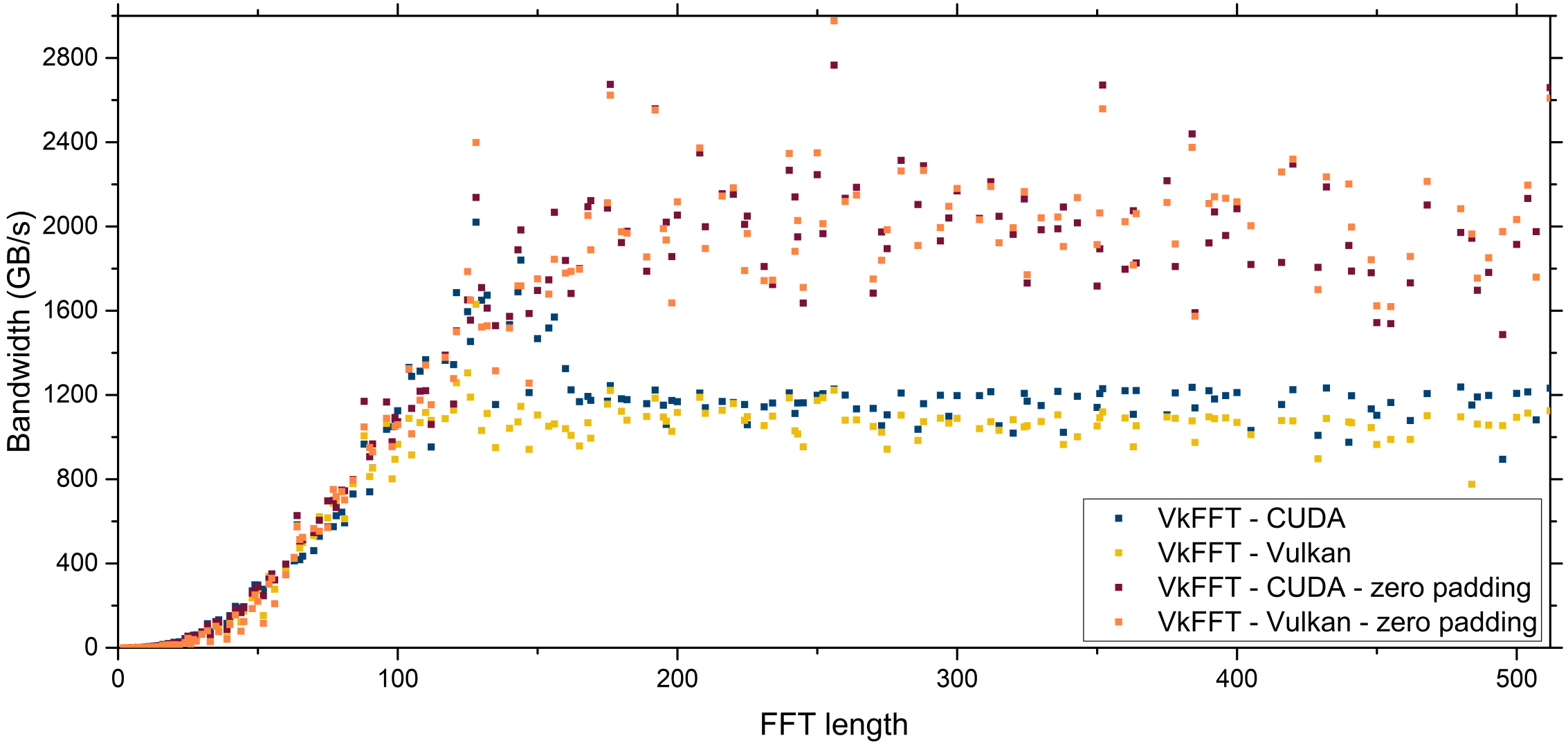
3D cube FP32 FFT on Nvidia A100



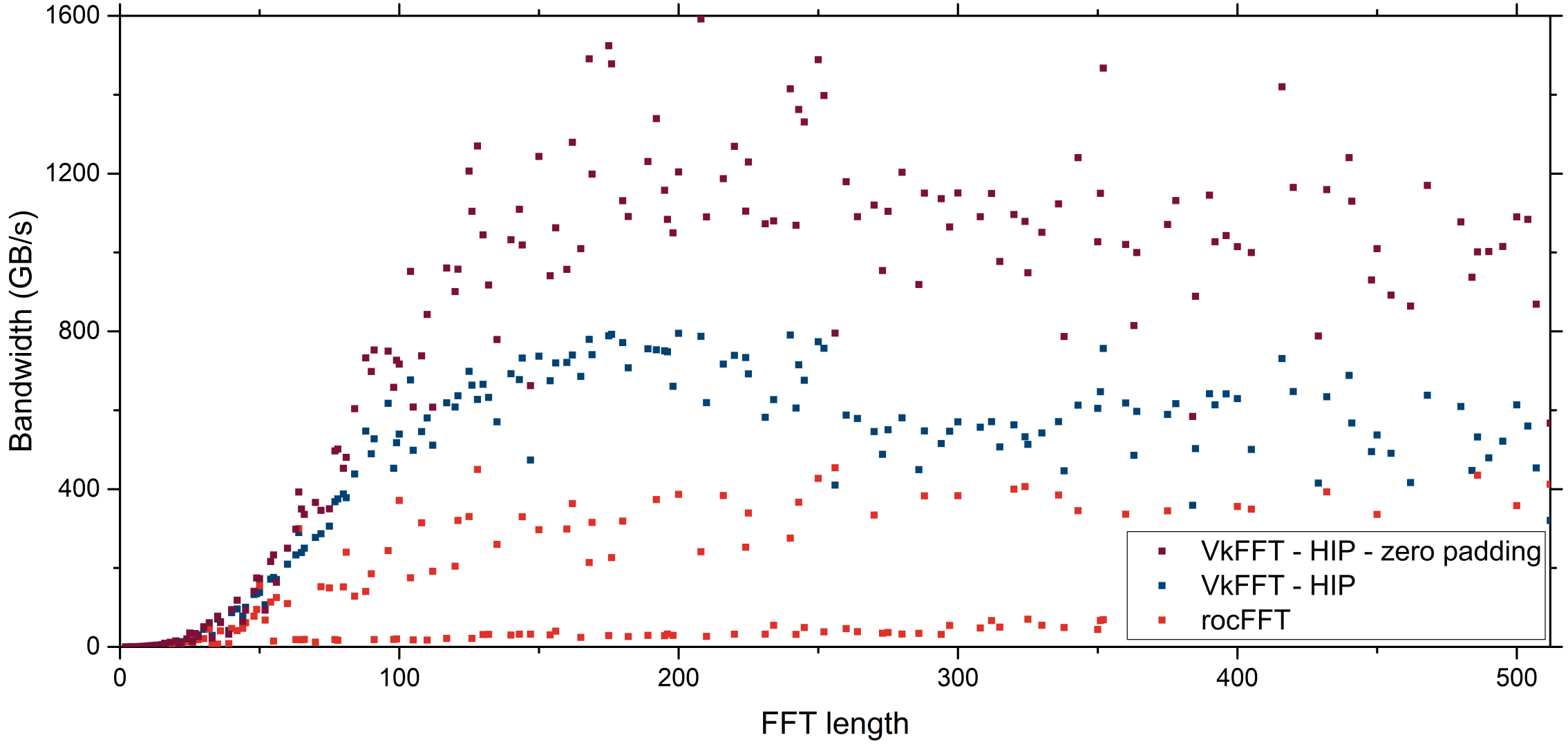
3D cube FP32 FFT on Nvidia A100



3D cube FP32 FFT on Nvidia A100



3D cube FP32 FFT on AMD MI100



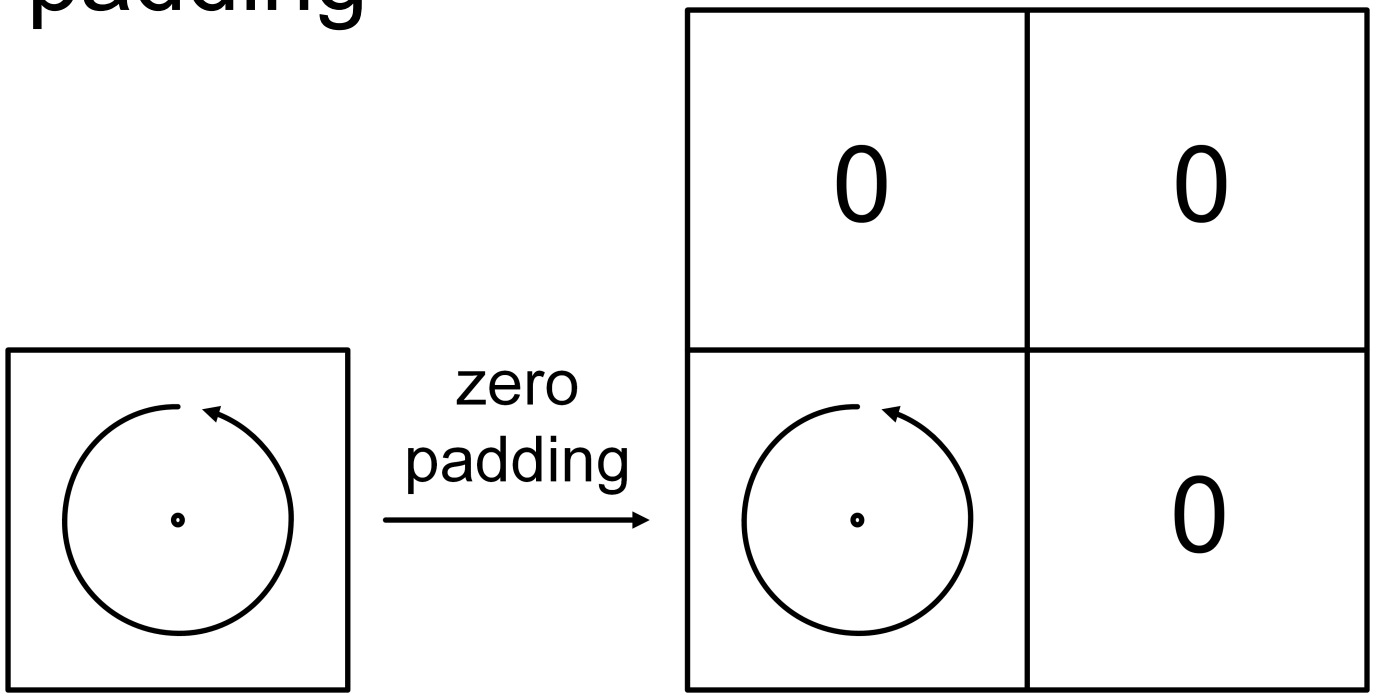
How is it possible to combine cross-platform support with optimized performance?

Answer: Metaprogramming

VkFFT metaprogramming paradygm

1. VkFFT algorithm is backend-abstracted
2. VkFFT generates platform optimized code at run-time
   - Vulkan – glslang compiler
   - CUDA – NVRTC
   - HIP – HIPRTC
   - OpenCL - native

Native zero padding



VkFFT omits sequences full of zeros and doesn't upload memory, known to be zero

- Up to 2x speed increase for 2D FFTs
- Up to 3x speed increase for 3D FFTs

Register overutilization

- Register file is usually much bigger (255KB) than available shared memory (32-64KB)
- Register overutilization: each thread stores its own part of the input sequence in its registers and uses shared memory to exchange data with other threads when it is needed
- Comes at a cost of reduced occupancy due to the increased register pressure

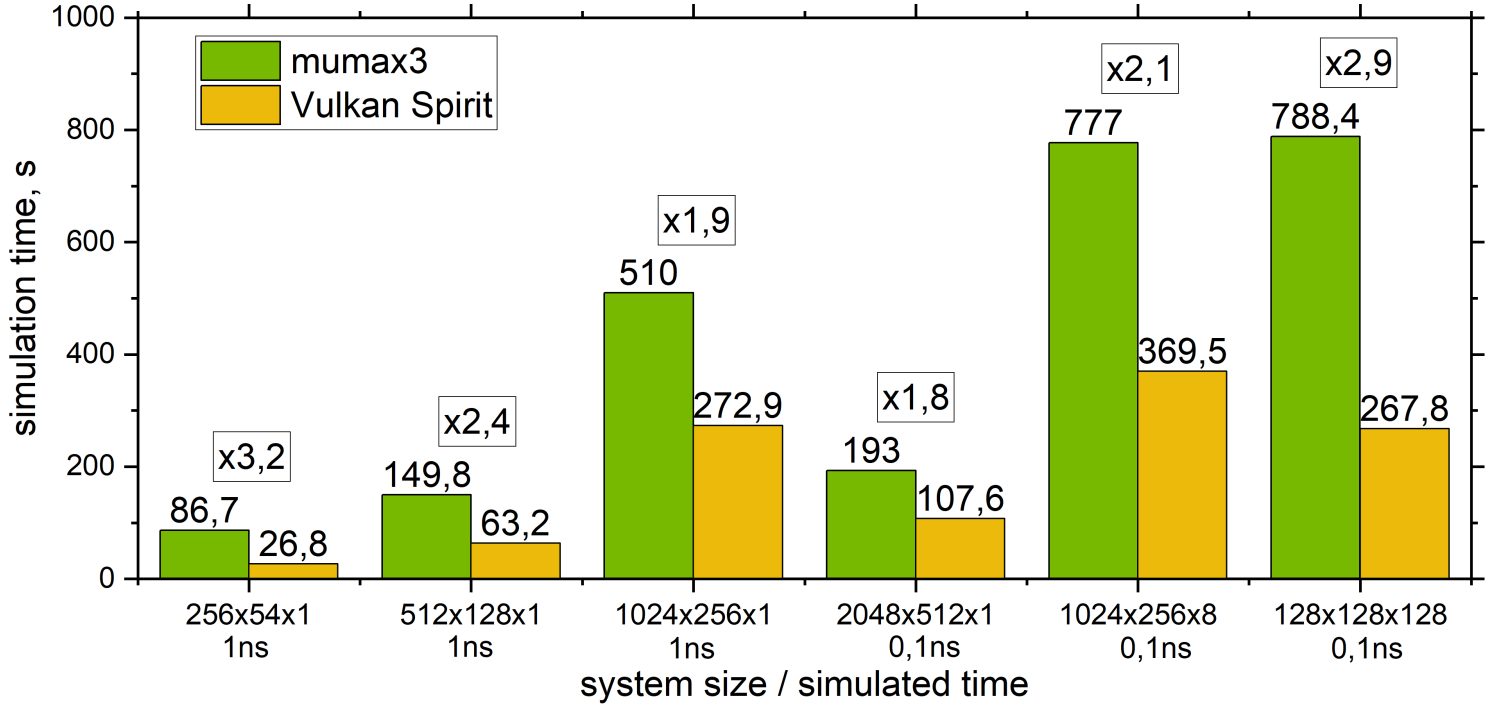
Merged convolution support

- Fourier transform of a convolution is the pointwise product of signals Fourier transforms (Convolution theorem)
- Merging of the last FFT step with kernel multiplication and the first step of inverse FFT provides substantial memory transfer savings
- FFTs of big sequences can be performed without data reordering – better locality

Note on VkFFT precision

- VkFFT precision is verified against FFTW to have similar orders of error ratio/difference as other GPU FFT libraries
- More information & tests on GitHub: https://github.com/DTolm/VkFFT

Vulkan Spirit: Micromagnetic simulations



Comparison of Vulkan Spirit and mumax3 on Nvidia GTX 1660 Ti

The standard problem four is chosen as the test system with RK4 solvers used in both software packages. Performance is measured in terms of time taken to complete 1ns (0.1ns for big systems) of simulation on open-boundary system with DDI. The fixed timestep is chosen to be 0.01ps.

GitHub: https://github.com/DTolm/spirit

VkResample: Vulkan FFT upscaler

GitHub: https://github.com/DTolm/VkResample

VkMatch: Real-time image registration on GPU

Reference

Input image

Match result

Medium post: https://towardsdatascience.com/real-time-image-registration-on-gpu-with-vkfft-library-c4e47f8050a0

Thank you for joining me today!

Check out VkFFT on: https://github.com/DTolm/VkFFT