HIGH-PERFORMANCE MULTI-GPU EEG SIGNAL PROCESSING

Bálint Tóth Zoltán Juhász

University of Pannonia,

Department of Electrical Engineering and Information Systems

AIM OF STUDY & MOTIVATION

- Implementation of EEG preprocessing steps that can be used on preexascale supercomputer architectures
- Selected DSP algorithms:
 - Convolution in time domain
 - Wavelet transform in time domain
- Main goals:
 - High computational performance
 - Scalability
 - Efficient use of the underlying GPU architecture

EEG SIGNAL PROCESSING

- Scientific EEG experiments in general
 - High temporal resolution (millisecond range)
 - High sampling rate: 1-30 kHz
 - Many channels (electrodes): 64 300
 - High amount of data: several GB per measurement
- Processing pipeline
 - Many complex DSP algorithms in sequence
- Time consuming on common hardware
 - Can take several hours for one subject
 - For group measurements, processing time can reach days



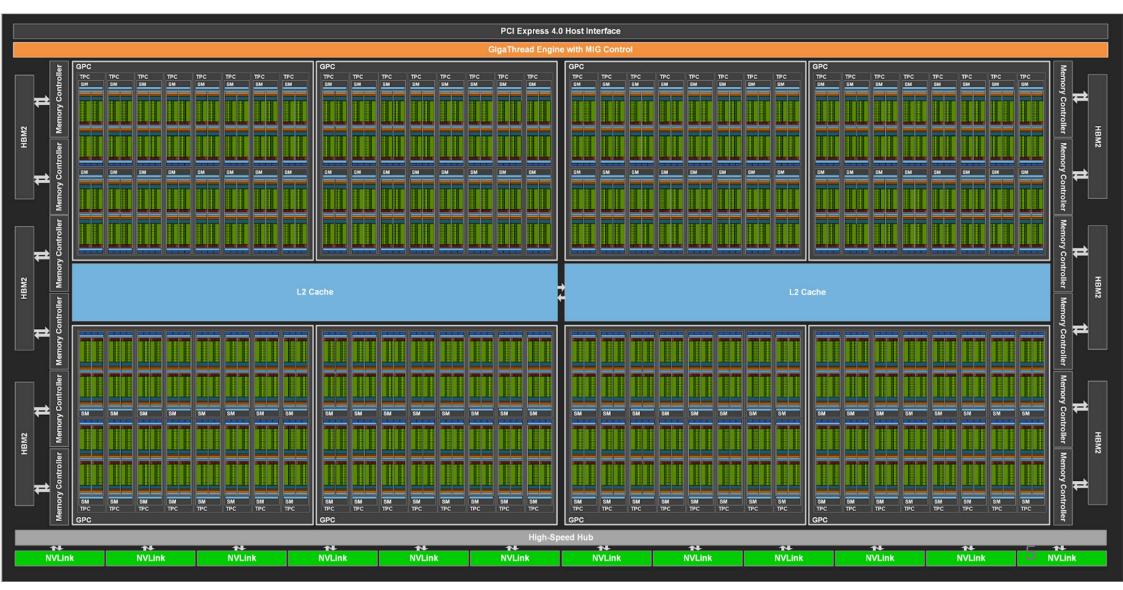
RELATED TECHNOLOGIES

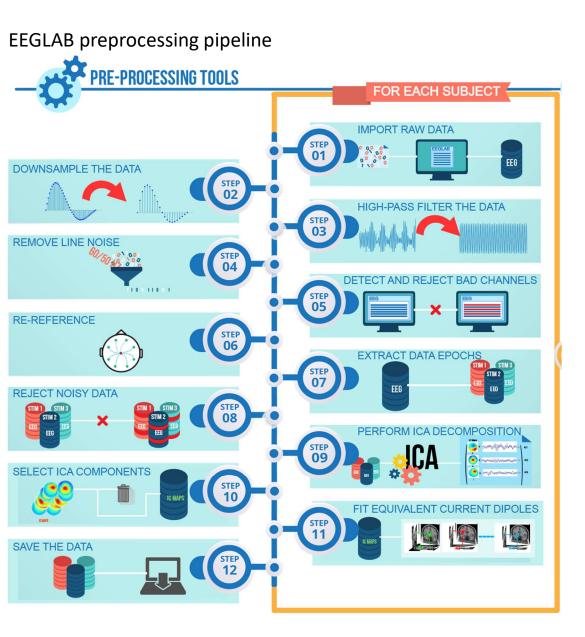
- NVIDIA CUDA
 - To enable extreme parallel computation
 - Hierarchical memory and thread model
- Message Passing Interface (MPI)
 - Communication for distributed memory systems
- NVIDIA A100 Datacentre GPU
 - CUDA Core count: 6912
 - SM count: 108
 - Shared memory size: 160 KB
 - Device memory size: 40GB
 - Device memory bandwidth: 1,55 GB/s
 - Performance of one GPU: 19,5 TFlop/s

	_	_	_	_	_	_	L1 Instru		n Cac	ne	_	_	_	_	_	_		
		L0 Ir	nstruct	ion C	ache							L0 In	struc	tion C	ache			
Warp Scheduler (32 thread/clk) Dispatch Unit (32 thread/clk)									Warp Scheduler (32 thread/clk)									
	Di	spatch	n Unit ((32 th	read/c	:lk)					Di	spatch	1 Unit	(32 th	read/o	:lk)		
	Reg	gister	File (1	6,384	4 x 32	-bit)					Reg	ister	File (1	16,384	4 x 32	!-bit)		
INT32 INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64				
INT32 INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64				
INT32 INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64				
INT32 INT32	IT32 INT32 FP32 FP32	FP32	FP64		TENSOR CORE				INT32 INT32		FP32	FP32 FP32		FP64		TENSOR CORE		
INT32 INT32	FP32	FP32	FP	64	TE	INSO	RCORE		INT32	INT32	FP32	FP32	FP	64	TE	INSO	RCORE	
INT32 INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64				
INT32 INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64				
INT32 INT32	FP32	FP32	FP	64					INT32	INT32	FP32	FP32	FP	64				
LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU		LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU	
-		1.0.1		ion C	acho			╎┌	_			1010		tion C	acho			
	Wa		nstruct Ieduler	_		/clk)		jĻ		_	Wai	_	istruci edule			/clk)		
		rp Sch	nstruct Ieduler 1 Unit ((32 t	hread							L0 In rp Sch spatch	edule	r (32 t	hread			
	Di	rp Sch spatci	eduler	(32 ti (32 th	hread/ read/c	:lk)					Di	rp Sch	edule 1 Unit	r (32 t (32 th	hread	:lk)		
INT32 INT32	Di Reç	rp Sch spatci	eduler 1 Unit ((32 th (32 th (6,384	hread/ read/c	:lk)			INT32	INT32	Di	rp Sch spatch jister	edule 1 Unit File (1	r (32 t (32 th	hread	:lk)		
INT32 INT32	Di Reg FP32	rp Sch spatch gister	eduler h Unit (File (1	(32 th (32 th (6,384 64	hread/ read/c	:lk)			INT32 INT32		Di Reg	rp Sch spatch jister FP32	edule 1 Unit File (1	r (32 th (32 th 16,384 64	hread	:lk)		
	Di Reg FP32 FP32	rp Sch spatch gister FP32	File (1	(32 th (32 th (6,384 64 64	hread/ read/c	:lk)				INT32	Di Reg FP32	rp Sch spatch jister FP32 FP32	edule 1 Unit File (1 FP FP	r (32 th (32 th 16,384 64	hread	:lk)		
INT32 INT32	Di Reg FP32 FP32 FP32	rp Sch spatch gister FP32 FP32	eduler 1 Unit (File (1 FP	(32 th (32 th (6,384 64 64 64	hread/ read/c 4 x 32	:lk) :-bit)			INT32	INT32 INT32	Di Reg FP32 FP32	rp Sch spatch jister FP32 FP32 FP32	edule 1 Unit File (1 FP FP	r (32 th (32 th 16,384 64 64 64	hread read/ 4 x 32	elk) 2-bit)		
INT32 INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32	rp Sch spatch jister FP32 FP32 FP32	eduler 1 Unit (File (1 FP FP	(32 th (32 th (6,384 64 64 64 64	hread/ read/c 4 x 32	:lk) :-bit)	R CORE		INT32	INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32	rp Sch spatch ister FP32 FP32 FP32 FP32	edule 1 Unit File (1 FP FP	r (32 th (32 th 16,38 64 64 64 64	hread read/ 4 x 32	elk) 2-bit)	R CORE	
INT32 INT32 INT32 INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32 FP32	rp Sch spatch gister FP32 FP32 FP32 FP32	eduler 1 Unit (File (1 FPI FPI FPI	(32 th (32 th (6,384 64 64 64 64 64	hread/ read/c 4 x 32	:lk) :-bit)	R CORE		INT32 INT32 INT32	INT32 INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32	rp Sch spatch ister FP32 FP32 FP32 FP32 FP32	edule Unit File (1 FP FP FP FP	r (32 th (32 th 16,38 64 64 64 64	hread read/ 4 x 32	elk) 2-bit)	R CORE	
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch gister FP32 FP32 FP32 FP32 FP32	File (1 File (1 FP FP FP FP	(32 th (32 th (32 th 64 64 64 64 64 64	hread/ read/c 4 x 32	:lk) :-bit)	R CORE		INT32 INT32 INT32 INT32	INT32 INT32 INT32 INT32 INT32	PP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch ister FP32 FP32 FP32 FP32 FP32 FP32	edule Unit File (1 FP FP FP FP	r (32 th (32 th 16, 38 64 64 64 64	hread read/ 4 x 32	elk) 2-bit)	R CORE	
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch gister FP32 FP32 FP32 FP32 FP32 FP32	File (1 File (1 FPI FPI FPI FPI	(32 th (32 th (32 th (6,384 64 64 64 64 64 64 64	hread/ read/c 4 x 32	:lk) :-bit)	R CORE		INT32 INT32 INT32 INT32 INT32	INT32 INT32 INT32 INT32 INT32 INT32	Dis Reg FP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch ister I FP32 FP32 FP32 FP32 FP32 FP32 FP32	edule Unit File (1 FP FP FP FP	r (32 th (32 th 16, 38) 64 64 64 64 64 64	hread read/ 4 x 32	elk) 2-bit)	R CORE	
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch gister FP32 FP32 FP32 FP32 FP32 FP32 FP32	File (1	(32 th (32 th (32 th (6,384 64 64 64 64 64 64 64	hread/ read/c 4 x 32	:lk) :-bit)	R CORE		INT32 INT32 INT32 INT32 INT32 INT32	INT32 INT32 INT32 INT32 INT32 INT32	Di Reg FP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch ister I FP32 FP32 FP32 FP32 FP32 FP32 FP32	edule Unit File (1 FP FP FP FP FP	r (32 th (32 th 16, 38) 64 64 64 64 64 64	hread read/ 4 x 32	elk) 2-bit)	R CORE	
INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch jister FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	File (1 File (1 FPI FPI FPI FPI FPI FPI FPI	(32 th (32 th 6,384 64 64 64 64 64 64 64 64 64 10/	hread/c 4 x 32 TE	Ik) I-bit) INSO			INT32 INT32 INT32 INT32 INT32 INT32 INT32 INT32	INT32 INT32 INT32 INT32 INT32 INT32 INT32 LD/ ST	Di Reg FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	rp Sch spatch FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	edule Unit File (1 FP FP FP FP FP FP FP	r (32 th (32 th 16,38- 64 64 64 64 64 64 64 64 64 64 10/	hread/4 4 x 32 TE	ENSO		

NVIDIA A100 Streaming Multiprocessor

NVIDIA A100 GPU ARCHITECTURE





EEG PREPROCESSING OPERATIONS

- Filtering
 - Convolution
 - Fourier transform

• Time-Frequency analysis

- Wavelet transform
- Hilbert transform
- Short Time Fourier transform

Spectral density estimation

- Fourier transform
- Wavelet transform
- Convolution

WAVELET TRANSFORM

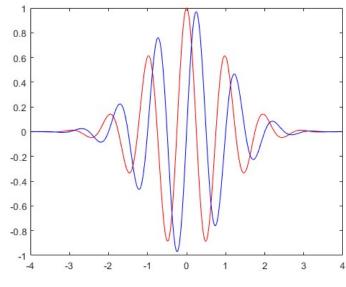
Operation

$$W\{s(t)\}(T,\tau) = \int_{-\infty}^{\infty} s(t)\Psi(\frac{t-\tau}{T}) dt$$

- Morlet basis-wavelet: $\Psi(t) = e^{j\omega_0 t} e^{-\frac{t^2}{2}}$
- In the case of multiple frequencies, the basis wavelet needs to be scaled
- Convolution for each frequency of each channel:

$$S[m] = \sum_{n=0}^{N-1} w_f[n] \cdot s[m-n]$$

• Filterbank structure



Morlet basis wavelet

BASE ALGORITHM

- Multi-channel signal
- Multiple frequency components
 - Increment: $\frac{f_{max}-f_{min}}{F}$
- Basis wavelet length depends on the sampling rate

$$O(C \cdot F \cdot (M + N - 1))$$

C – number of channels M – data points in one channel F – number of frequency components N – length of the basis wavelet

```
for c := 0 to C do
for f := fmin to F do
    for m := 0 to M+N-1 do
        re := 0.0
        im := 0.0
        for n := 0 to N-1 do
            re += w_re[c,f,n] * s[c,m-n]
            im += w_im[c,f,n] * s[c,m-n]
            result[c,f,m] :=
                  sqrt(re*re + im*im)
        end
    end
end
```

end

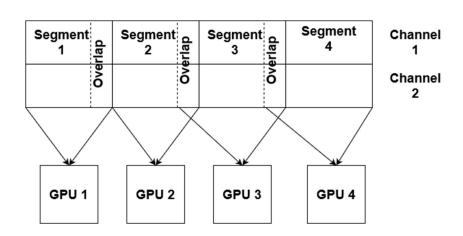
SINGLE-GPU IMPLEMENTATION

- Thread hierarchy:
 - 2D input, 3D output
 - 2D Grid with row-major mapping
 - Each element in the result matrix is mapped to a thread
 - Channels are mapped to rows
 - Spectra for each frequency components are mapped to columns
- Memory usage:
 - Basis wavelet too long for shared memory end
 - Wavelet filter bank is generated on the GPU
 - Avoiding unnecessary CPU usage and host device copy

end end

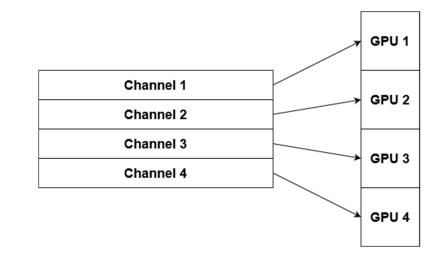
MULTI-GPU COMMUNICATION SCHEMES

- Main goal: minimise the amount of data that needs to be sent between GPUs
- CUDA-aware MPI
 - Data can be sent directly in between GPU cards' DRAM with the same API calls
 - Displacement buffer corruption for large inputs
 - Manual collective communication
- Asymmetric distribution scheme
 - Whole measurement data is divided into equal-length segments
 - Requires overlap for numerical correctness
 - Number of GPU cards is not strictly bound by the problem size
 - Works well for convolution



MULTI-GPU COMMUNICATION SCHEMES

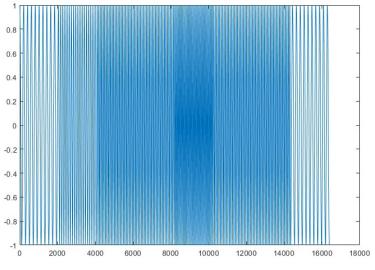
- Channel based distribution scheme
 - Channels are assigned equally to GPUs
 - Number of GPUs is bound by the number of channels
 - Less unnecessary data transfers for long convolution windows
- Feasible option for wavelet transform
 - Kernel can be modified for single channel computation resulting in a more efficient solution
 - For group measurements, data from multiple subjects can be processed simultaneously



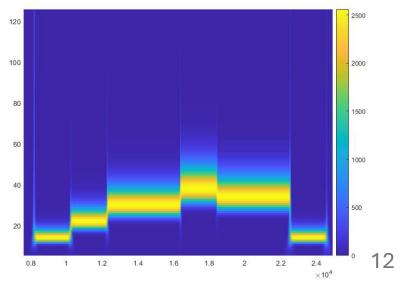
MEASUREMENT RESULTS

- Devices used:
 - NVIDIA GeForce RTX 3050 individual kernel performance measurements
 - Komondor supercomputer (1 16 NVIDIA A100 GPU) – multi-GPU runtime measurements
- Test data
 - 64 128 channels
 - 2 million data points per channel
 - Approximately 15 minutes of 2KHz EEG measurement data
 - 65536 data points (30 seconds) in case of Wavelet transform
 - Randomly generated vector
- MATLAB Signal Processing Toolbox was used to verify the correctness of the implementations

Example of a frequency-modulated test signal



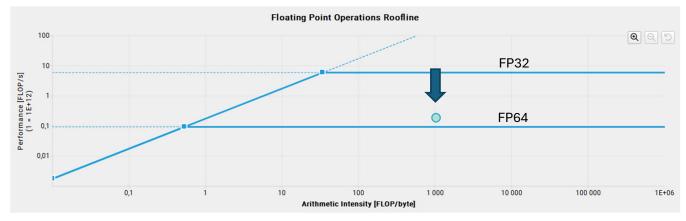
Scalogram of the test signal



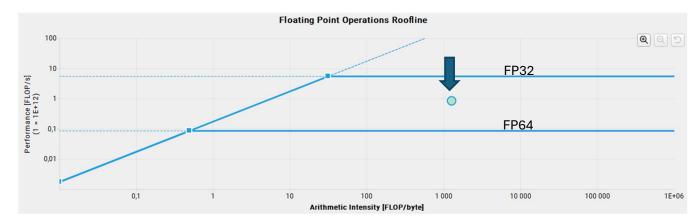
WAVELET TRANSFORM KERNEL PERFORMANCE

- Roofline model:
 - Performance in relation to arithmetic intensity
- Wavelet kernel is compute bound
- Performance increases by adding more channels

FP32 and FP64 roofline plot for single channel



FP32 and FP64 roofline plot for 1024 channels

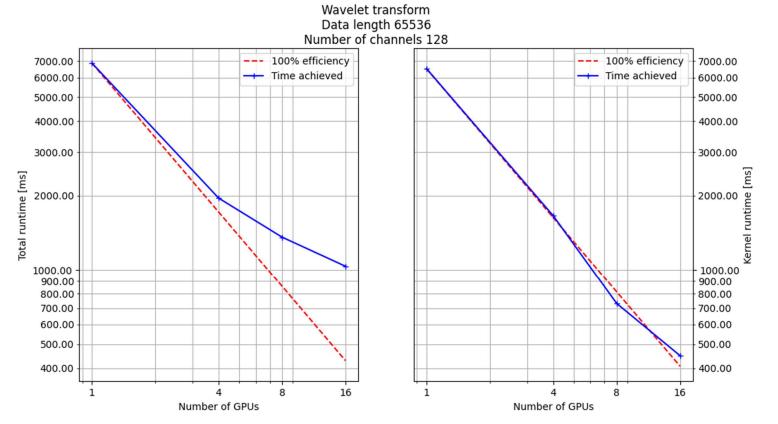


WAVELET TRANSFORM SCALABILITY

Total runtime:

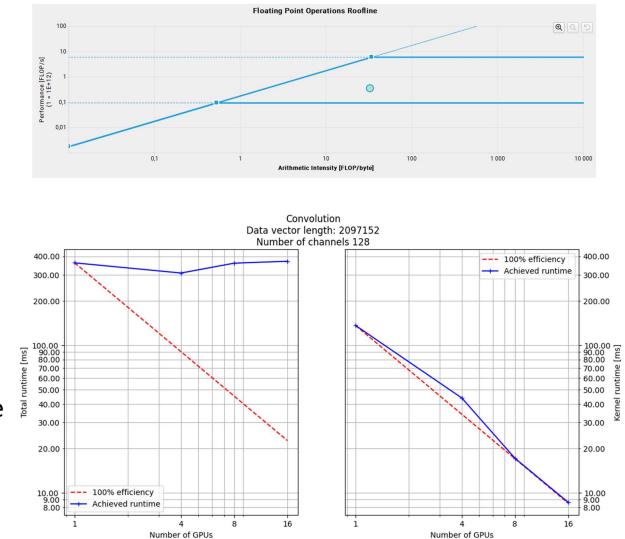
- From the start of distribution
- To the end of collection

- Kernel runtime decreases with more GPUs
- The size of the task allows strong scaling
- In case of the multi-GPU implementation, computation time is dominant
- Communication time is a near-constant 1 second



CONVOLUTION

- Kernel is on the edge of memory boundedness
- Performance increases by adding more channels
- Kernel runtime for one GPU decreases proportionally to the number of GPU cards
- Communication time is dominant
- The algorithm is scalable, it can be used as part of a pipeline



15

SUMMARY

- Conclusion:
 - Two working DSP algorithms designed for single-GPU use
 - Different communication schemes for multi-GPU use
 - Both algorithms can be scaled efficiently in supercomputer environments
 - MPI is a serious scalability bottleneck for large problem sizes
- Further development:
 - Elimination of MPI in favour of direct GPU-GPU communication with NVIDIA frameworks and distributed file system reading
 - Integration to the rest of the EEG preprocessing pipeline