Simulation of Pulse Propagation in Nonlinear Optical Fibers Using GPUs

G. Kuracz, L. Reina Kiperman, F. Reyna, and P. I. Fierens

The problem

Pulse propagation in non-linear optical fibers → no analytical solution

A = Pulse envelope

$$\frac{\partial A}{\partial z} = \mathbf{L}A + \mathbf{N}A$$

Linear operator: (*Fourier transforms*)

$$\mathbf{L} = -\frac{1}{2}\sum_{k\geq 0}\frac{i^k}{k!}\alpha_k\frac{\partial^k}{\partial T^k} + \sum_{k\geq 2}\frac{i^{k+1}}{k!}\beta_k\frac{\partial^k}{\partial T^k}$$

Nonlinear operator

$$\mathbf{N}A = \sum_{k\geq 0}\frac{i^{k+1}}{k!}\gamma_k\frac{\partial^k}{\partial T^k}\left(A\int_{-\infty}^{+\infty}R(T')|A(z,T-T')|^2dT'\right)$$

Dudley, Genty & Coen, RMP (2006)

Runge-Kutta 4 – Interaction Picture

State-of-the-art algorithm

Change of variables: $A^{\mathrm{IP}}(z,T) = \exp\{-(z-h/2)\mathbf{L}\}A(z,T)$

$$\frac{\partial A^{\mathrm{IP}}}{\partial z} = \breve{\mathbf{N}}A^{\mathrm{IP}} = \mathrm{e}^{-\left(z-\frac{h}{2}\right)\mathbf{L}}\mathbf{N}\mathrm{e}^{+\left(z-\frac{h}{2}\right)\mathbf{L}}A^{\mathrm{IP}}.$$

The linear operator is eliminated → A^{IP} can be integrated with, e.g., RK4

Hult, Journal Lightwave Tech. (2007)

Runge-Kutta 4 – Interaction Picture

$z_{k+1} = z_k + h,\ a_k(T)$: approximation to $A(z_k, T)$

$$a_k^{\mathrm{IP}} = \mathrm{e}^{\frac{\mathbf{L}h}{2}} a_k,$$

$$K_1 = \mathrm{e}^{\frac{\mathbf{L}h}{2}} \mathbf{N} a_k, \quad K_2 = \mathbf{N}\left(a_k^{\mathrm{IP}} + \frac{h}{2} K_1\right), \quad K_3 = \mathbf{N}\left(a_k^{\mathrm{IP}} + \frac{h}{2} K_2\right), \quad K_4' = \mathbf{N}\mathrm{e}^{\frac{\mathbf{L}h}{2}}\left(a_k^{\mathrm{IP}} + hK_3\right)$$

$$a_{k+1} = \mathrm{e}^{\frac{\mathbf{L}h}{2}}\left[a_k^{\mathrm{IP}} + \frac{h}{6}(K_1 + 2K_2 + 2K_3)\right] + \frac{h}{6} K_4'$$

8 FFTs

8 FFTs for the convolutions

Hult, Journal Lightwave Tech. (2007)

Runge-Kutta 4 – Interaction Picture

- 16 FFTs ~ $16 \times N \times \log_2 N$ products per step
- ~ $15 \times N$ other products (depends on details of the implementation)
- $N \geq 15 \Rightarrow \gtrsim 10^7$ products per step

Can some of these operations be parallelized?

GPUs

- Allow parallelization of operations into several processing cores

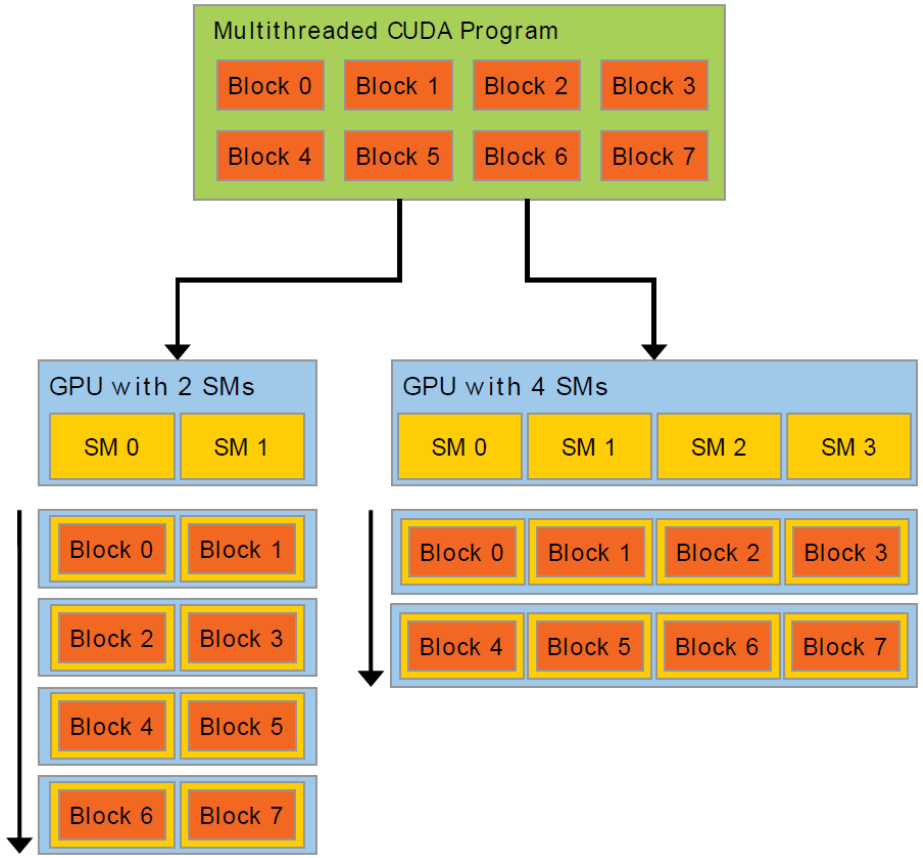
- Reported speedups of FFTs of 3x – 100x (with respect to a PC)

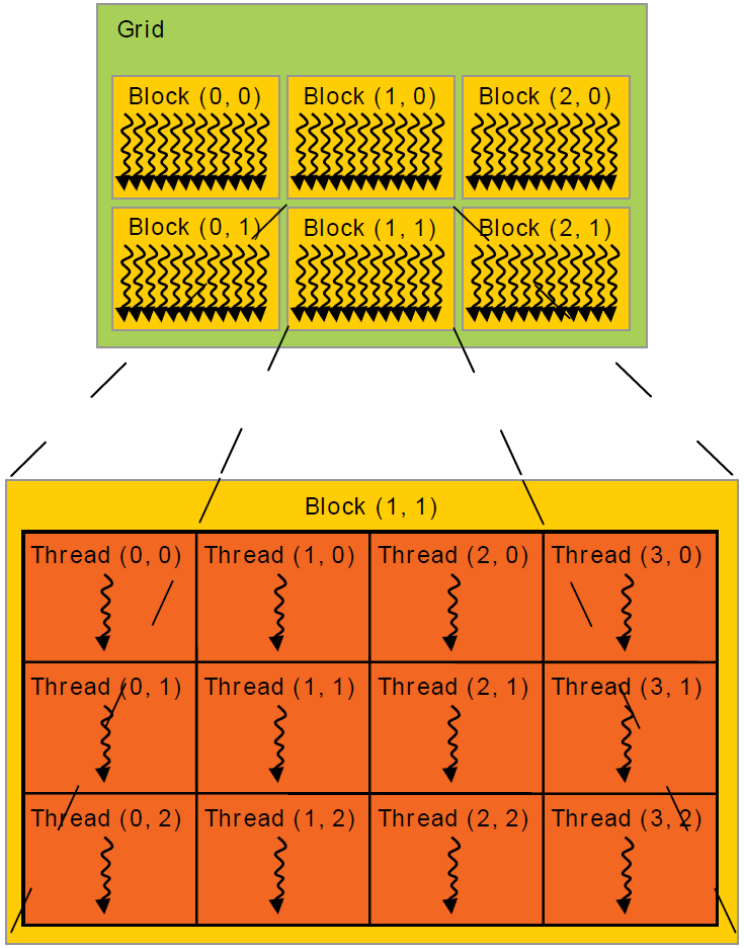
  Volkov & Kazian, (2008)
  Lee et al., SIGARCH (2010)

- One of the keys: minimization of data transfers between CPU RAM & GPU RAM

Nvidia processors & CUDA

- CUDA: a parallel programming model invented by Nvidia





Nvidia, (2015)

Nvidia processors & CUDA

- Simple loop parallelization

Function

```
__global__ void cuda_mainLoop_for04(cufftDoubleComplex *vin_dev,
                                    cufftDoubleComplex *k4_dev,
                                    const unsigned int nt,
                                    const double dzsixth)
{
    const unsigned int k = blockIdx.x*blockDim.x + threadIdx.x;
    if (k < nt)
    {
        vin_dev[k].x += k4_dev[k].x*dzsixth;
        vin_dev[k].y += k4_dev[k].y*dzsixth;
    }
}
```

Call

```
cuda_mainLoop_for04 <<< cnt, t >>> (vin_dev, k4_dev, nt, dzsixth);
```

Nvidia processors & CUDA

- cuFFT: Simple API, similar to that of FFTW (Fastest Fourier Transform on the West)

Mem Alloc

```
cufftDoubleComplex *vin_dev, *vout_dev;

cudaStatus = cudaMalloc((void**)&vin_dev,vin_sizeVar);
cudaStatus = cudaMalloc((void**)&vout_dev,vout_sizeVar);
```

- One of the keys: minimization of data transfers between CPU RAM & GPU RAM

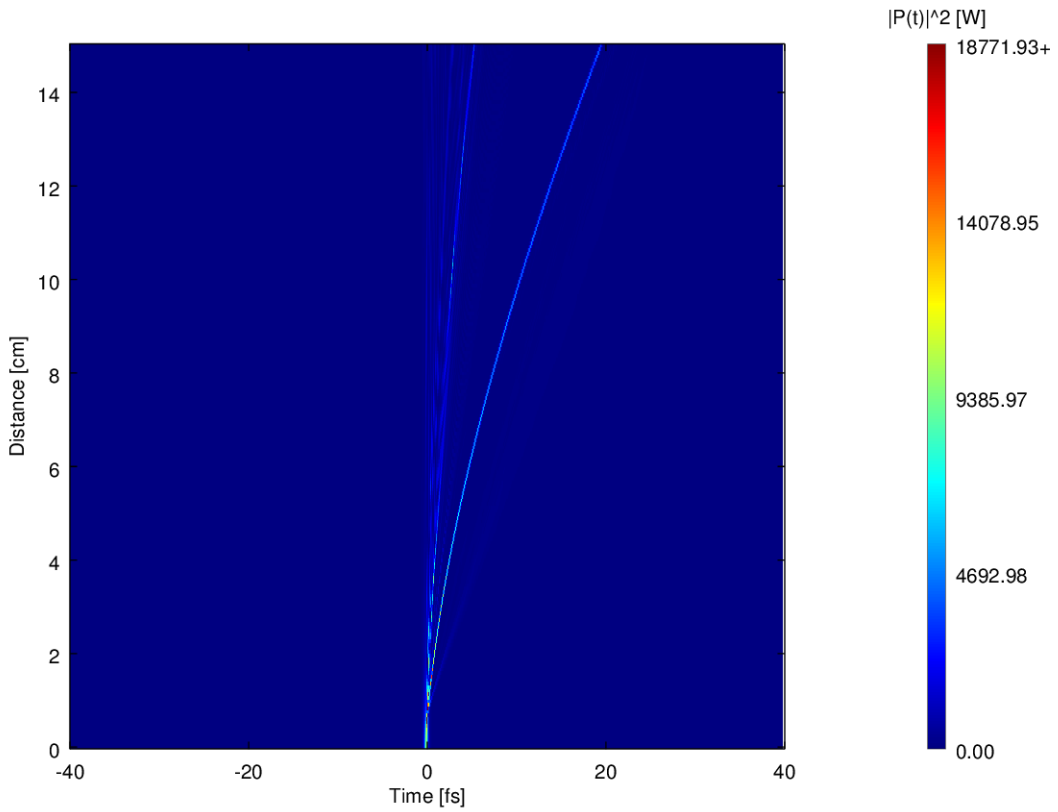
Plan

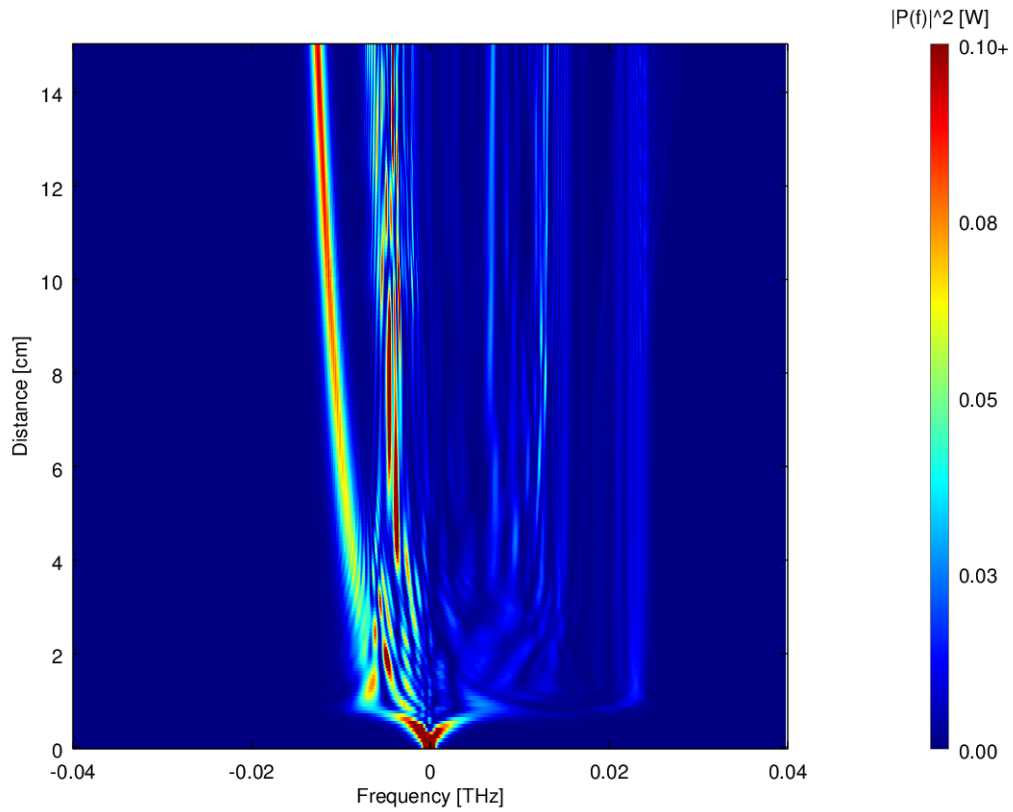
```
cufftHandle  pbw;
cufftStatus = cufftPlan1d(&pbw, nt, CUFFT_Z2Z, 1);
```

Execution

```
cufftExecZ2Z(pbw,vin_dev,vout_dev,CUFFT_INVERSE);
```

Example


|P(t)|^2 [W]
18771.93+
14078.95
9385.97
4692.98
0.00
Distance [cm]
0
2
4
6
8
10
12
14
-40
-20
0
20
40
Time [fs]


|P(f)|^2 [W]
0.10+
0.08
0.05
0.03
0.00
Distance [cm]
0
2
4
6
8
10
12
14
-0.04
-0.02
0
0.02
0.04
Frequency [THz]

Performance

Characteristic	Computer 1	Computer 2
Motherboard	Asus M3N78-VM	Asus S1A55-M LX
Processor	AMD Phenom II X4 945 Processor, 3GHz, 8 MB Cache	AMD A8-3870, 3GHz Quad Core, 4 MB Cache
Hard Disk	500GB 7200 RPM SATA II Western Digital Caviar Blue	500Gb 7200 RPM SATA III Seagate
RAM	8 GB DDR2 800 MHz	16 GB DDR3 800 MHz
OS	Ubuntu 14.04	Windows 7 SP1
GPU	Zotac GeForce GTX 960 amp	TESLA K40c
CUDA	Version 7.5	Version 7

Performance

Characteristic	Zotac GeForce GTX 960*	TESLA K40c◊
CUDA cores	1024	2880
Memory size	2 GB (GDDR5)	12 GB (GDDR5)
Multiprocessors	8	15
Memory Clock	3505 MHz	3004 MHz
Memory Bus Width	128-bit	384-bit

* Financial support of Banco Santander Río through the ITBA's R+D Initiation Contest allowed the its acquisition and funded other parts of the project

◊ Donated by NVidia

Performance

Execution time in seconds/speedup

N	FFTW	Zotac GeForce GTX 960		TESLA K40c	
2^{13}	6.6	8.5	0.8x	9.3	0.7x
2^{14}	66.0	12.9	5.1x	11.7	5.6x
2^{17}	1425.0	103.2	13.8x	46.7	30.5x
2^{20}	15126.1	Didn't fit in memory		301.4	50.2x

Conclusions

- Parallelization of loops and efficient implementations of FFTs allow up to 50x speedups in the simulation of pulse propagation in optical fibers

- It is convenient to perform all computations on the GPU memory

  → Simulation may not fit in the memory of some devices

- No improvement is obtained for small simulations

- RK4-IP is not intrinsically parallelizable: are there any alternative algorithms which may be better suited for parallelization in GPUs?