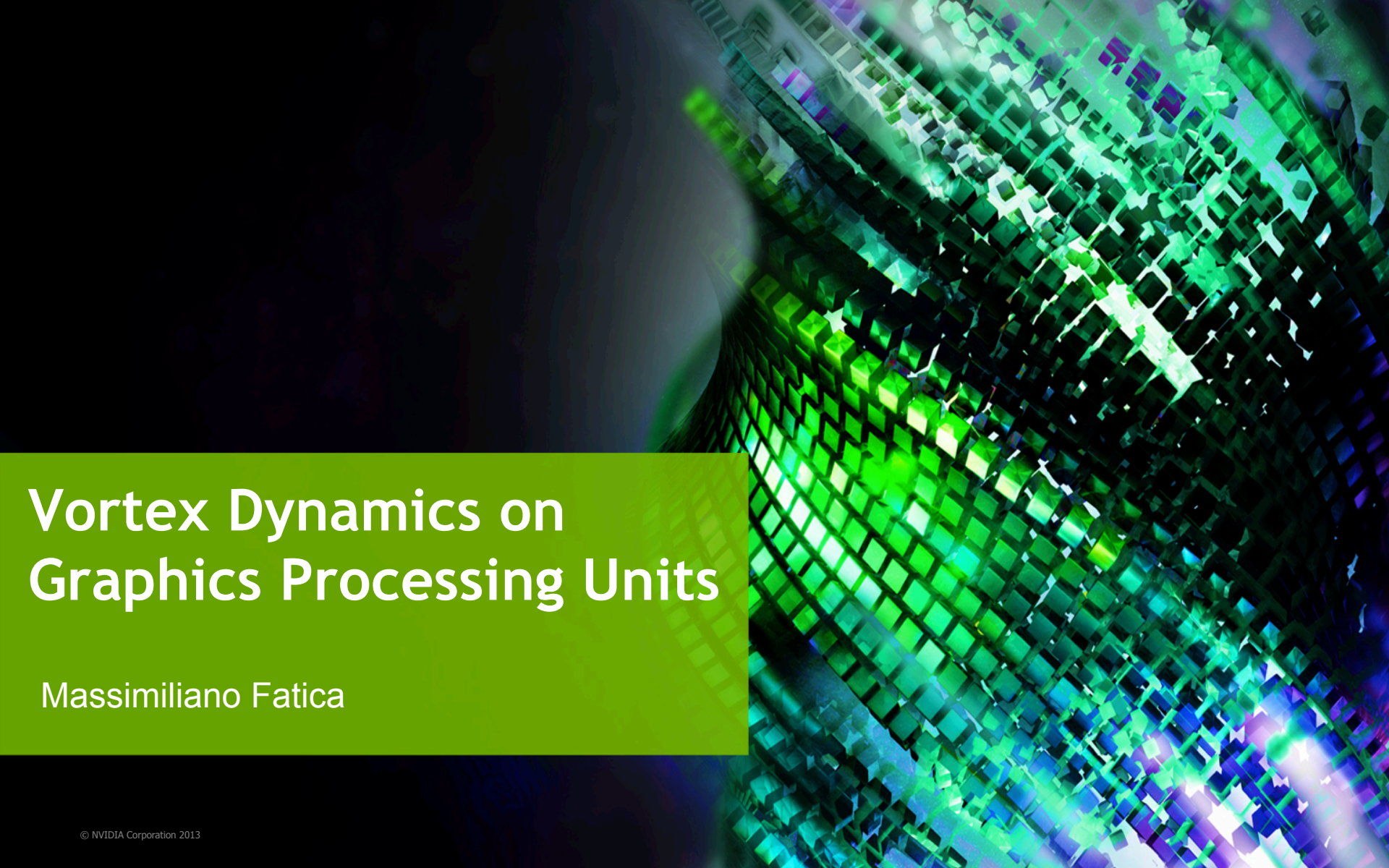


Vortex Dynamics on Graphics Processing Units

Massimiliano Fatica

GPU Computing

GPUs are very attractive in High Performance Computing:

- Massive multithreaded many-core chips
- High flops count (both SP and DP)
- High memory bandwidth, ECC
- Programming languages: CUDA C, CUDA Fortran, OpenACC, Python, OpenCL, MATLAB
- Tools: debuggers, profilers, libraries (BLAS, FFT, LAPACK, Totalview, DDT, Nvidia Visual Profiler)

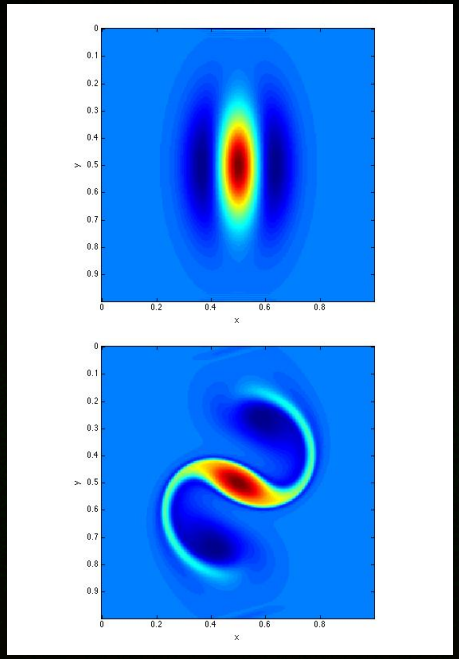
Growing adoption in computational fluid dynamics:

- Senocak, Perot,..

Case study

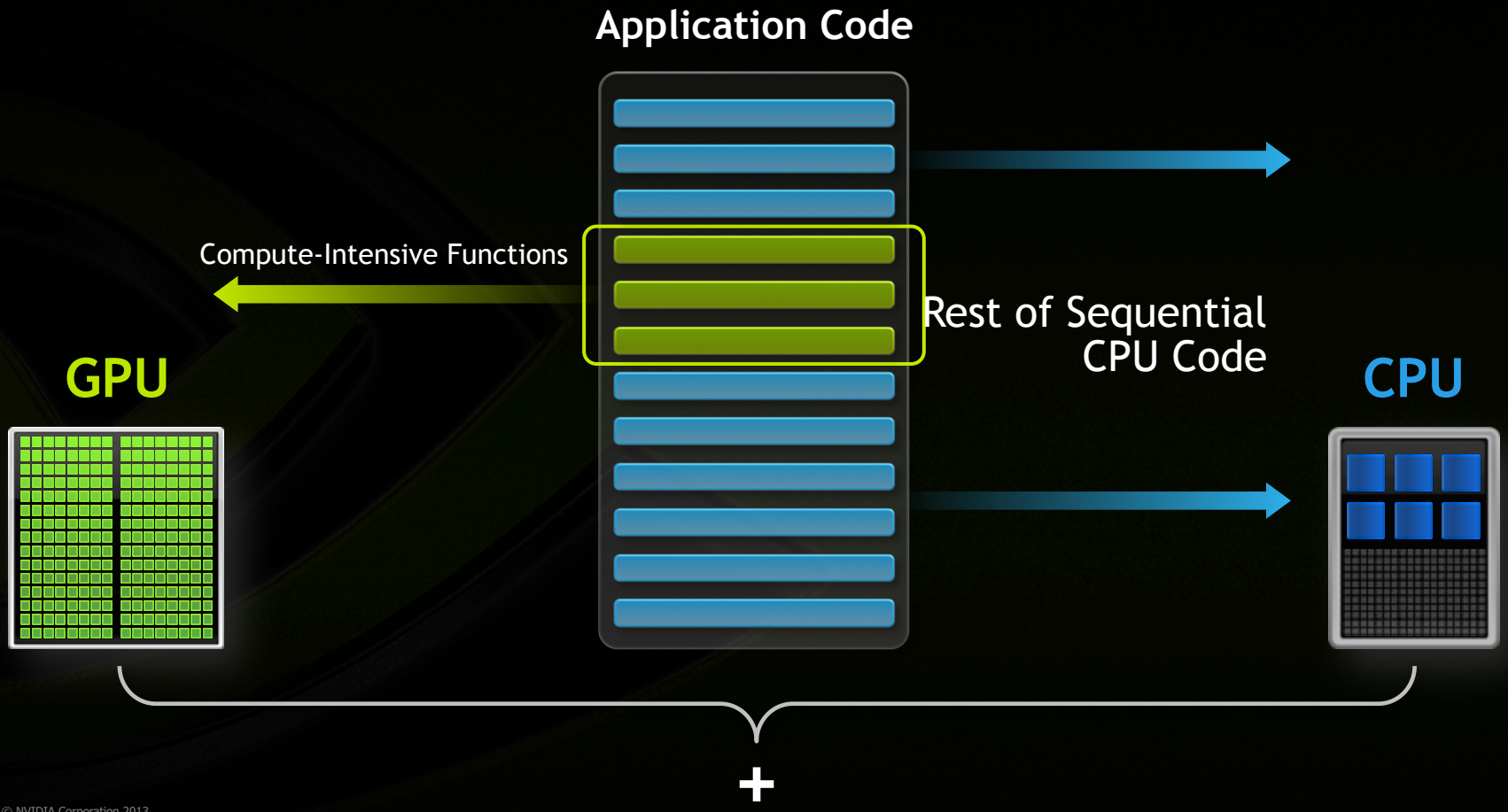
Fourier pseudo-spectral simulation of 2D incompressible inviscid flow:

- Vorticity-stream function formulation:
  - Poisson equation to compute stream function from vorticity
- 2D periodic domain
- RK4 time advancement
- Hyper-viscosity



Matlab scripts from Univ. of Washington (Amath 571) for evolution of isolated elliptic vortex and 2D isotropic turbulence

How GPU Acceleration Works



CUDA C: C with a few keywords

```
void saxpy_serial(int n, float a, float *x, float *y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}
// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

Standard C Code

```
__global__ void saxpy_parallel(int n, float a, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n)  y[i] = a*x[i] + y[i];
}
// Invoke parallel SAXPY kernel with 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

Parallel C Code

CUDA & MATLAB

Several ways to use GPUs:

- GPU-enabled MATLAB functions in several toolboxes:

```
A = gpuArray(rand(2^16,1));
B = fft(A);
```

- CUDA kernel integration in MATLAB applications

- MEX interface

Mex files for CUDA/Octave

A typical mex file will perform the following steps:

1. Allocate memory on the GPU
2. Transfer the data from the host to the GPU
3. Rearrange the data layout for complex data if needed
4. Perform computation on GPU (library, custom code)
5. Rearrange the data layout for complex data
6. Transfer results from the GPU to the host
7. Clean up memory and return results to MATLAB

Vorticity source term

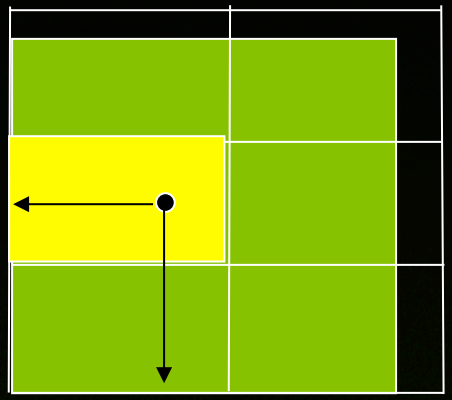
```
function S = Szeta(zeta,k,nu4)

% Pseudospectral calculation of vorticity source term
%  S = -(- psi_y*zeta_x + psi_x*zeta_y) + nu4*del^4 zeta
% on a square periodic domain, where zeta = psi_xx + psi_yy is an NxN matrix
% of vorticity and k is vector of Fourier wavenumbers in each direction.
% Output is an NxN matrix of S at all pseudospectral gridpoints

  zetahat = fft2(zeta);
  [KX KY]  = meshgrid(k,k); % Matrix of (x,y) wavenumbers corresponding to Fourier mode (m,n)
  del2 = -(KX.^2 + KY.^2);
  del2(1,1) = 1;  % Set to nonzero to avoid division by zero when inverting Laplacian to get psi
  psihat = zetahat./del2;
  dpsidx = real(ifft2(1i*KX.*psihat));
  dpsidy = real(ifft2(1i*KY.*psihat));
  dzetadx = real(ifft2(1i*KX.*zetahat));
  dzetady = real(ifft2(1i*KY.*zetahat));
  del2max = norm(del2(:),inf);
  diff4 = real(ifft2(del2.^2.*zetahat));
  S = -(-dpsidy.*dzetadx + dpsidx.*dzetady) - nu4*diff4;
```

Poisson solver

```
__global__ void solve_poisson (cufftComplex *c, float *kx, float *ky, int N)
{
     int idx   = blockIdx.x*blockDim.x+threadIdx.x;
     int idy   = blockIdx.y*blockDim.y+threadIdx.y;
    // use shared memory to minimize multiple access to same k values
     __shared__ float kx_s[BLOCK_WIDTH], ky_s[BLOCK_HEIGHT]
      if (threadIx.x < 1) kx_s[threadIdx.x] = kx[idx];
      if (threadIx.y < 1) ky_s[threadIdx.y] = ky[idy];
      __syncthreads();
    if ( idx < N && idy <N){
            unsigned int index = idx +idy*N;
            float scale =  - 1.f/ ( kx_s[threadIdx.x]*kx_s[threadIdx.x]
                                  + ky_s[threadIdy.y]*ky_s[threadIdy.y] );
            if ( idx ==0 && idy == 0 ) scale =1.f;
             c[index].x *= scale;
             c[index].y*= scale;
            }
}
```



$$\hat{\phi} = -\frac{\hat{r}}{(k_x^2 + k_y^2)}$$

Results on laptop

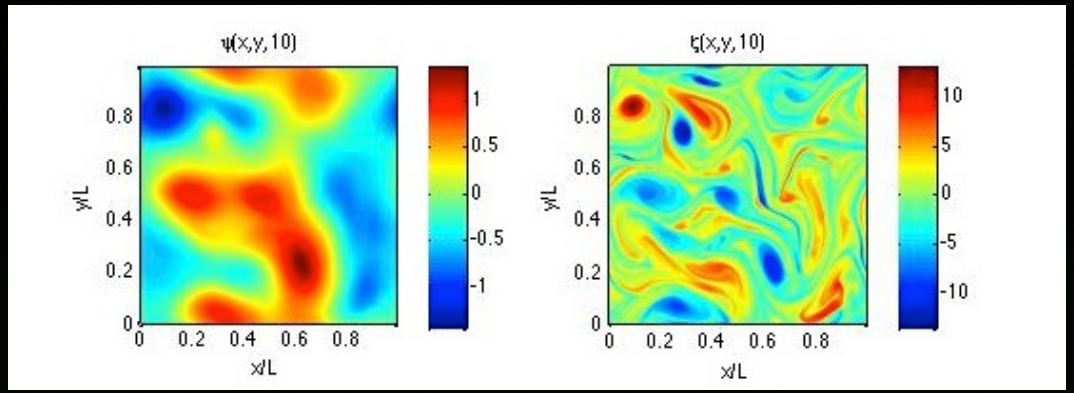
N	CPU	GPU
256	17s	10s
512	189s	76s

Elliptic Vortex

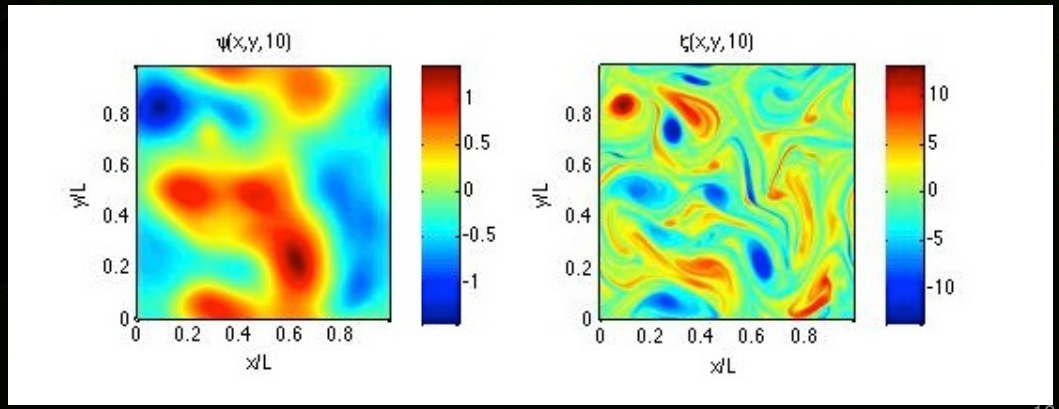
N	CPU	GPU
256	72s	43s
512	791s	351s

2D turbulence

CPU



GPU



Conclusions

Happy Birthday Prof. Orlandi and wishes for many more turbulent years !!!