Stanford University

CME 253A

Introduction to High Performance Computing and Parallel (GPU) Computing

Stanford Summer Session 2

26 June 2019 | Y2E2 111

Ludovic Räss

Stanford University, Department of Geophysics

lraess@stanford.edu

SIGMA

Session 2 - GPU computing

Today's agenda

- Lecture: GPU computing and mazama GPU server

- Programming: 1/ 1D acoustic wave - Matlab vs CUDA C
  2/ Connect to mazama and run a C executable
  3/ 1D to 2D acoustic wave in CUDA C

- Tasks: 1/ 2D elastic wave
  2/ 2D viscous
  3/ towards 3D elastic or viscous

Session 2 - GPU computing

Today's agenda

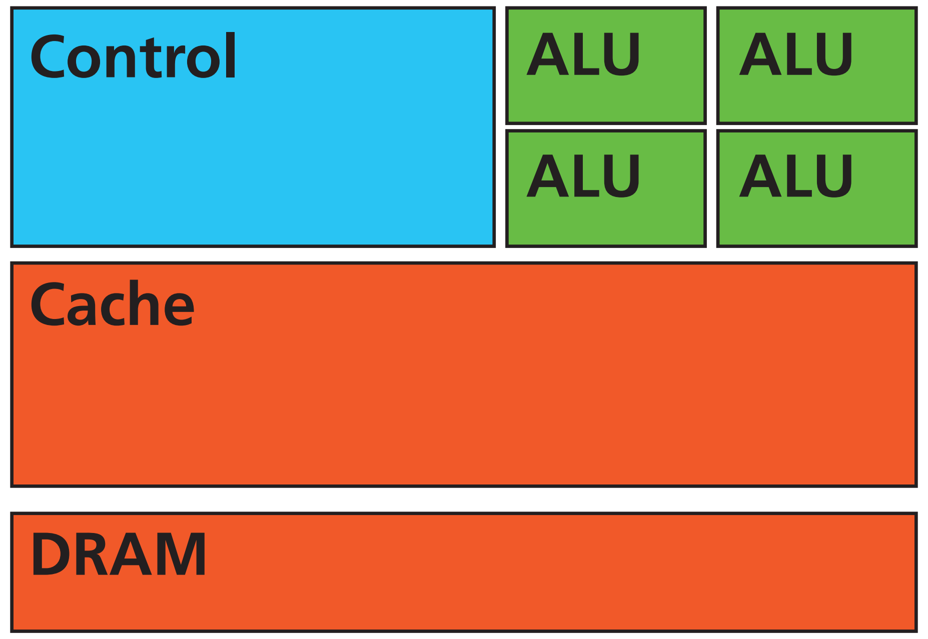
- Lecture: GPU computing and mazama GPU server

- Programming: 1/ 1D acoustic wave - Matlab vs CUDA C
  2/ Connect to mazama and run a C executable
  3/ 1D to 2D acoustic wave in CUDA C

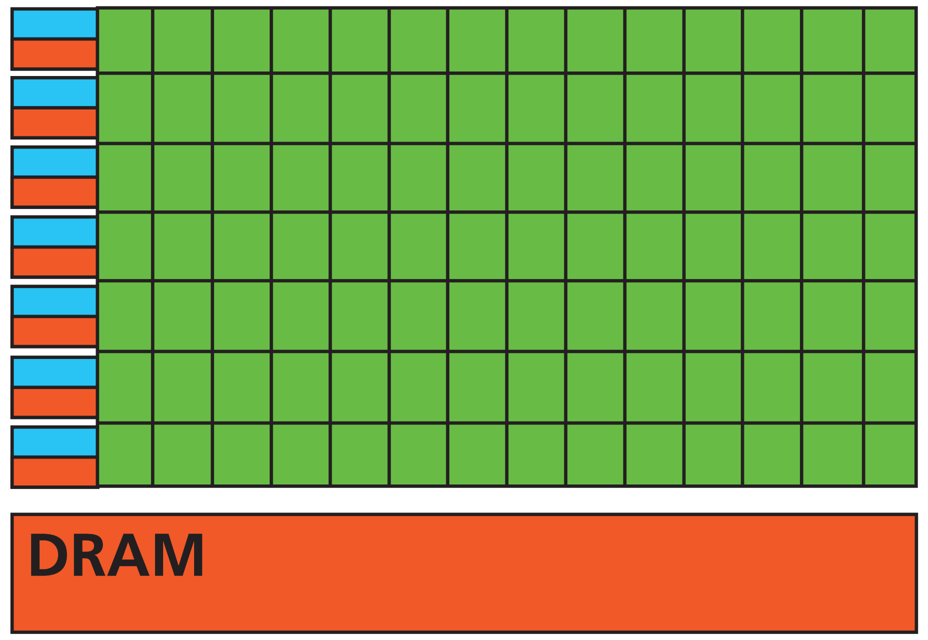
- Tasks: 1/ 2D elastic wave
  2/ 2D viscous
  3/ towards 3D elastic or viscous

GPU computing

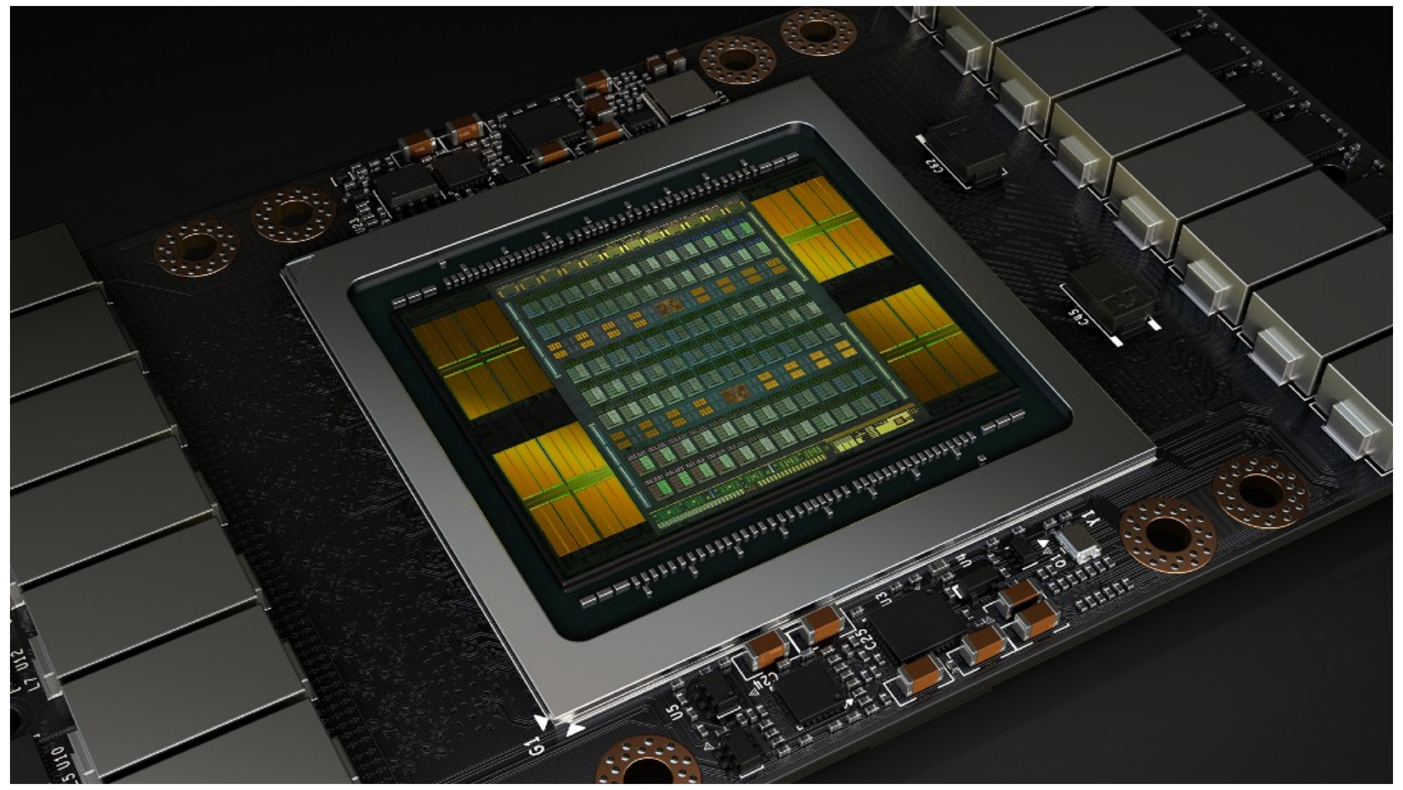
- Graphical Processing Unit (to paint the screen)
- Many small cores all doing the same job
- Large chip, close to memory
- High memory bandwidth - important for PDEs !



CPU

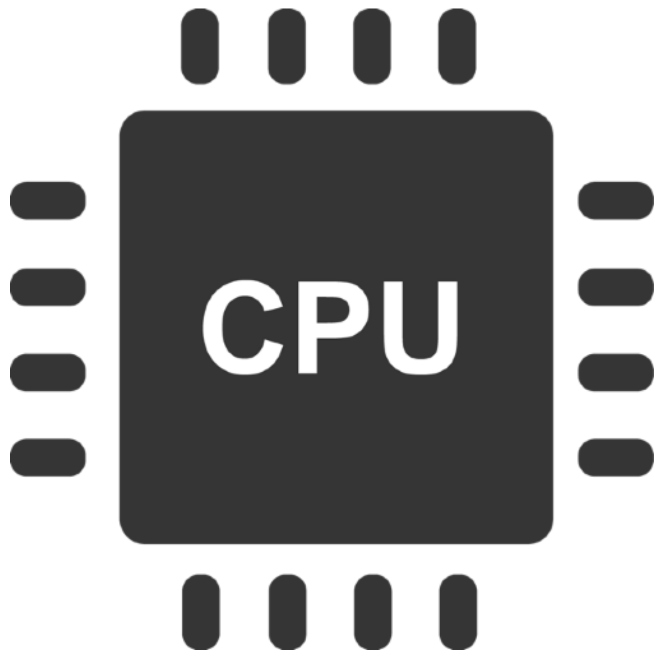


GPU



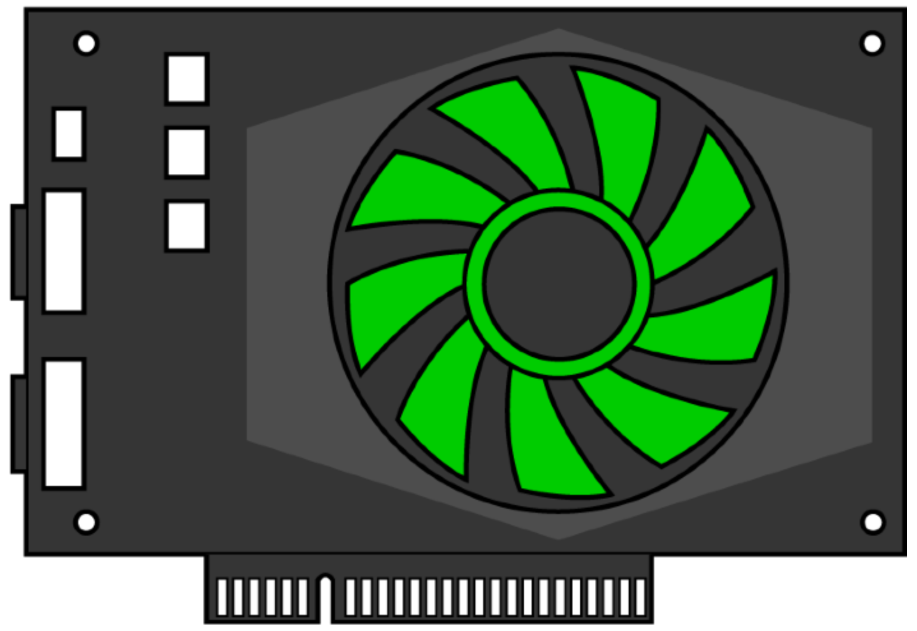
Why to use a GPU

- Serial vs parallel execution > loop vs vectorised code
- Simultaneous calculations > increased concurrency
- Order of magnitude higher memory bandwidth & flop/s count



0.5 TFlop/s

50 GB/s



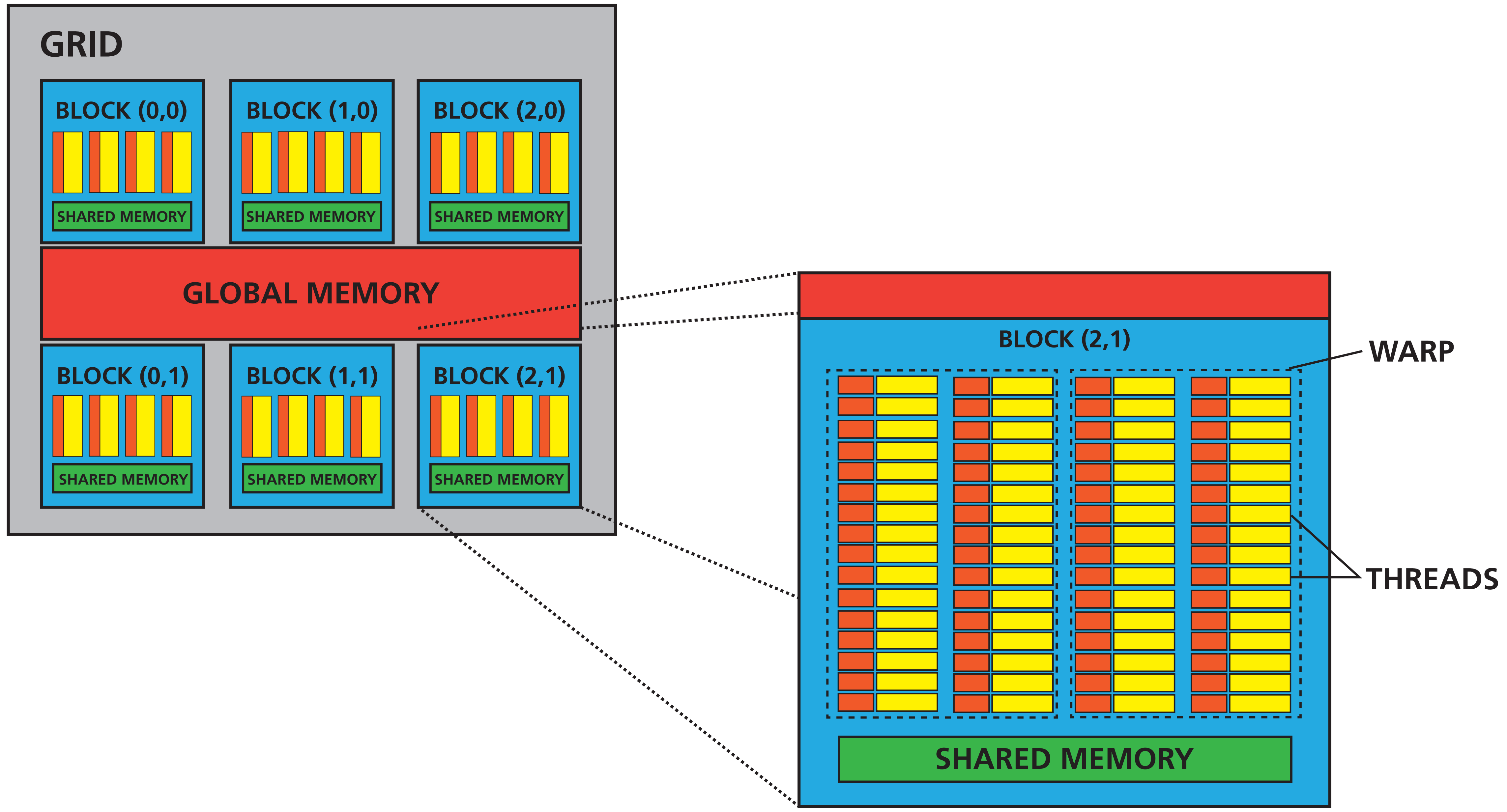
15 TFlop/s

700 GB/s

- Supercomputing available on your desk

Programming the GPU: CUDA

- CUDA: provides vectorised indexes to C / C++ for parallel execution
- Building elements: **`thread, block, grid`**
- Max. 1024 threads per block !
- Assign one thread per grid point
- All threads read the same code
  loop bounds > if statements



Session 2 - GPU computing

Today’s agenda

- Lecture: GPU computing and mazama GPU server

- Programming: 1/ 1D acoustic wave - Matlab vs CUDA C
  2/ Connect to mazama and run a C executable
  3/ 1D to 2D acoustic wave in CUDA C

- Tasks: 1/ 2D elastic wave
  2/ 2D viscous
  3/ towards 3D elastic or viscous

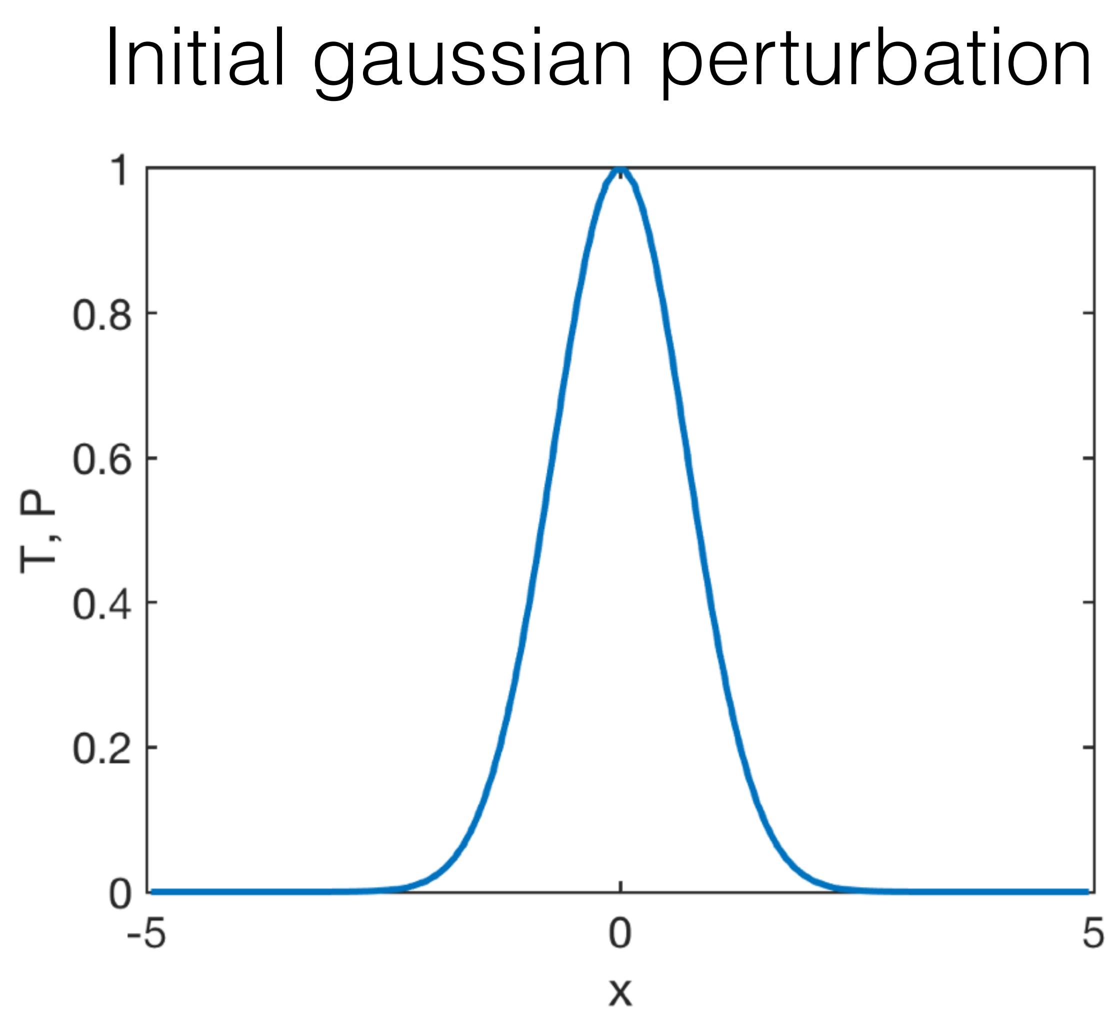
1/ 1D acoustic wave - Matlab vs CUDA C

Acoustic wave

$$\frac{1}{k}\frac{\partial P}{\partial t} = -\frac{\partial v_x}{\partial x}$$

$$\rho\frac{\partial v_x}{\partial t} = -\frac{\partial P}{\partial x}$$

```
% Physics
Lx  = 10;
k   = 1;
rho = 1;
% Numerics
nx  = 100;
dx  = Lx/nx;
nt  = 200;
dt  = dx/sqrt(k/rho)/2.1;
x   = (-Lx+dx)/2:dx:(Lx-dx)/2;
% Initial conditions
P   = exp(-x.^2);
V   = [0*P 0];
```



1/ 1D acoustic wave - Matlab vs CUDA C

- Minimal required changes:

  1/ Define some rules (macro) `#define`

  2/ Define `GPU_ID`

  3/ Define the domain size based on `grid = thread*block`

  4/ Copy memory from host (RAM) to device global memory

  5/ Define the function to run on the GPU `__global__ void`

  5/ Pass `block` and `grid` as argument to the GPU function

  7/ Gather the memory back from the device to host and visualise

- Compile using the Nvidia compiler `nvcc`

1/ 1D acoustic wave - Matlab vs CUDA C

- Define host (`_h`) and device (`_d`) pointers
- Initialise host: `A##_h = (DAT*)malloc((N)*sizeof(DAT));`

```
for(i=0; i<N; i++){A##_h[i]=(DAT)0.0;
```

- Allocate device pointer: `cudaMalloc(&A##_d,N*sizeof(DAT));`
- Copy host to device:

```
cudaMemcpy(A##_d,A##_h,N*sizeof(DAT),cudaMemcpyHostToDevice);
```

- Gather results:

```
cudaMemcpy(A##_h,A##_d,N*sizeof(DAT),cudaMemcpyDeviceToHost);
```

1/ 1D acoustic wave - Matlab vs CUDA C

- Initialise vectorised grid: `dim3 grid, block;`
  Where: `block.x=BLOCK_X; grid.x=GRID_X;`

- Select the GPU:
  ```
  gpu_id = GPU_ID; cudaSetDevice(gpu_id);
  cudaGetDevice(&gpu_id); cudaDeviceReset();
  cudaDeviceSetCacheConfig(cudaFuncCachePreferL1);
  ```

- Reserve threads (workers) for ghost nodes: `OVERLENGTH`

- Define resolution based on gird, block: `nx = BLOCK_X*GRID_X-OVERLENGTH;`

1/ 1D acoustic wave - Matlab vs CUDA C

- GPU function (kernel) `__global__ void` type
- Thread index in dimension x: `int ix = blockIdx.x*blockDim.x+threadIdx.x;`
- GPU function (kernel) and device synchronisation:
  `init<<<grid,block>>>(…); cudaDeviceSynchronize();`
- Free memory host and device: `free(A##_h); cudaFree(A##_d);`

2/ Connect to mazama-gpu

- CEES cluster GPU ressources: cees-mazama-gpu-2 and gpu-3 node
- You may need to be connected to the Stanford VPN
- Connect: `ssh <SUNetID>@cees-mazama-gpu-3`
  `or ssh <SUNetID>@cees-mazama-gpu-2` then `ssh cees-mazama-gpu-3`
- Go to scratch: `cd /scratch`
- Create a directory with your login name: `mkdir <SUNetID>`
- Copy the GPU code from your machine to mazama: scp <source> <target>
  `scp wave_1D.cu lraess@cees-mazama-gpu-2:/scratch/lraess`

2/ Connect to mazama-gpu

- /!\ `/scratch` is not backed-up. Copy your data back to your laptop or save it on data: `cd /data/cees` then `mkdir <SUNetID>`

- There is no job scheduler on mazama-gpu server nodes
  > Always check available resources prior to run your job:
  - CPU utilisation: `top`
  - GPU utilisation: `nvidia-smi`

- Compile a GPU code using the `nvcc` compiler:
  For the Nvidia Tesla P100: `nvcc -arch=sm_70 -O3 mycode.cu`

- Execute the generated output executable: `./a.out`

3/ 1D to 2D acoustic wave in CUDA C

- Use linear 1D indexing in 2D and 3D:

1D: `P(ix)     => P[ix]`
2D: `P(ix,iy) => P[ix+iy*nx]`

- Check bounds with if statements in order to exit from not active threads

Matlab loop version	CUDA C version

<pre>Vx = zeros((nx+1)*ny,1);
for iyM=1:ny, iy=iyM-1;
    for ix=2:nx
        Vx(ix+iy*(nx+1)) = …;
    end
end</pre>
	<pre>zeros(Vx ,nx+1,ny);
	if (iy<ny && ix>0 && ix<nx){
	        Vx[ix + iy*(nx+1)] = …;
	}</pre>

Session 2 - GPU computing

Today's agenda

- Lecture: GPU computing and mazama GPU server

- Programming: 1/ 1D acoustic wave - Matlab vs CUDA C
  2/ Connect to mazama and run a C executable
  3/ 1D to 2D acoustic wave in CUDA C

- Tasks: 1/ 2D elastic wave
  2/ 2D viscous (+add vertical gravity force)
  3/ towards 3D elastic or viscous

Task 1/ 2D acoustic to elastic wave

Acoustic waves	Elastic waves
$\frac{1}{k}\frac{\partial P}{\partial t} = -\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)$	$\frac{1}{k}\frac{\partial P}{\partial t} = -\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)$
$\rho\frac{\partial v_x}{\partial t} = -\frac{\partial P}{\partial x}$	$\rho\frac{\partial v_x}{\partial t} = -\frac{\partial P}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y}$
$\rho\frac{\partial v_y}{\partial t} = -\frac{\partial P}{\partial y}$	$\rho\frac{\partial v_y}{\partial t} = -\frac{\partial P}{\partial y} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{xy}}{\partial x}$
	$\frac{\partial \tau_{xx}}{\partial t} = 2G\left(\frac{\partial v_x}{\partial x} - \frac{1}{3}\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)\right)$
	$\frac{\partial \tau_{yy}}{\partial t} = 2G\left(\frac{\partial v_y}{\partial y} - \frac{1}{3}\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)\right)$
	$\frac{\partial \tau_{xy}}{\partial t} = 2G\frac{1}{2}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right)$

Task 2/ 2D elastic to viscous

Elastic waves	Viscous flow
$\frac{1}{k}\frac{\partial P}{\partial t} = -\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)$	$\frac{1}{k}\frac{\partial P}{\partial t} = -\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)$
$\rho\frac{\partial v_x}{\partial t} = -\frac{\partial P}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y}$	$\rho\frac{\partial v_x}{\partial t} = -\frac{\partial P}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y}$
$\rho\frac{\partial v_y}{\partial t} = -\frac{\partial P}{\partial y} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{xy}}{\partial x}$	$\rho\frac{\partial v_y}{\partial t} = -\frac{\partial P}{\partial y} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{xy}}{\partial x}$
$\frac{\partial \tau_{xx}}{\partial t} = 2G\left(\frac{\partial v_x}{\partial x} - \frac{1}{3}\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)\right)$	$\tau_{xx} = 2\eta\left(\frac{\partial v_x}{\partial x} - \frac{1}{3}\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)\right)$
$\frac{\partial \tau_{yy}}{\partial t} = 2G\left(\frac{\partial v_y}{\partial y} - \frac{1}{3}\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)\right)$	$\tau_{yy} = 2\eta\left(\frac{\partial v_y}{\partial y} - \frac{1}{3}\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right)\right)$
$\frac{\partial \tau_{xy}}{\partial t} = 2G\frac{1}{2}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right)$	$\tau_{xy} = 2\eta\frac{1}{2}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right)$

Outlook - Session 3

- Topic: Performance evaluation

- Programming: Time measurement, GB/s vs Flops/s, effective memory throughput

- Tasks: 2D and 3D elastic waves or viscous Stokes

- Mandatory for session 3: Matlab code in both loop and vectorised style. CUDA C GPU code reproducing Matlab results.

That’s it for today

- wp.unil.ch/geocomputing/
- lraess@stanford.edu