Laplace's equation – MPI + CUDA

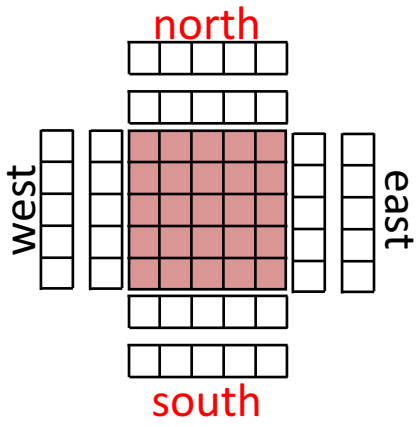
Data distribution

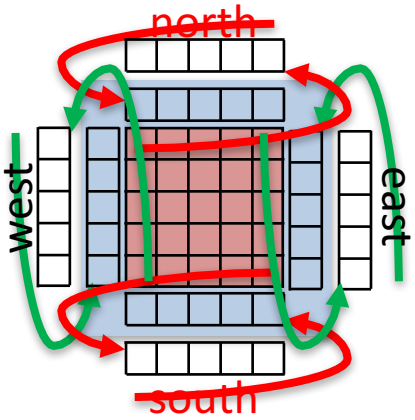
Create datatypes

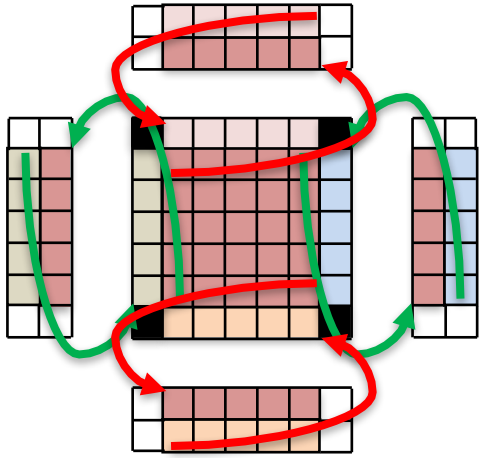
Exchange data with neighbors (north, south, east, west)

Do local computation

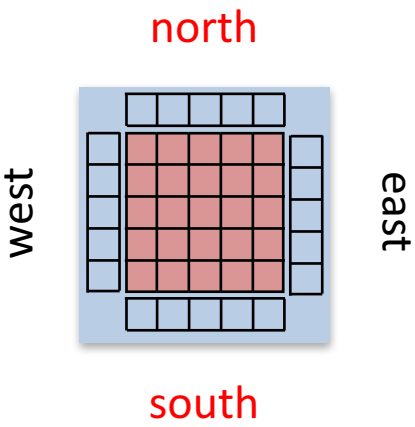
- Lets first assume we don't use datatypes (instead we will manually pack/unpack)

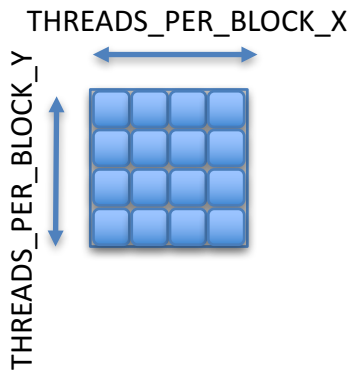






Programming CUDA





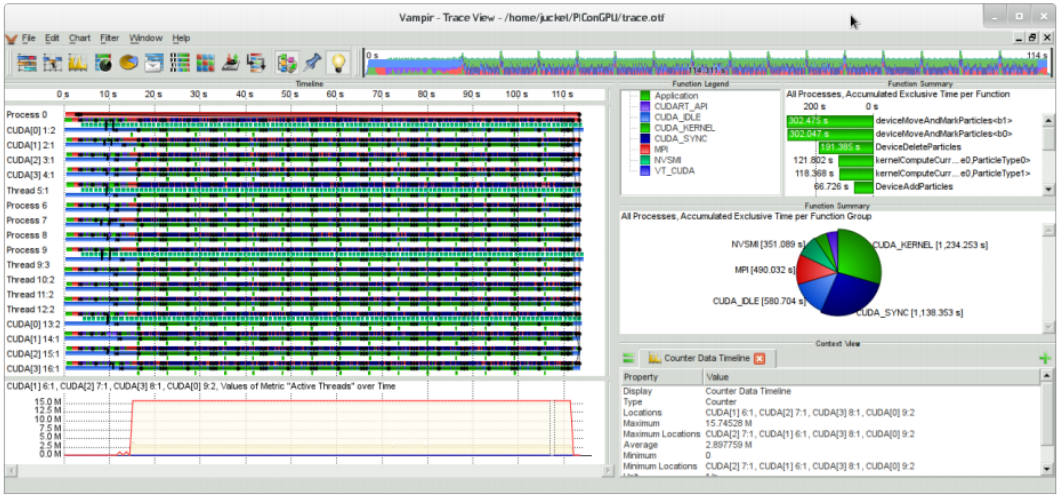
- Each little box is a light computational thread
- Going back to the data distribution methods (k-cyclic), the CUDA work distribution can be assimilated to a <X-cyclic, Y-cyclic> distribution
  - The goal is to evenly distribute the computational work over all threads available on the GPU
  - Warp: a group of 32 parallel threads, that executes exactly the same thing (but are named distinguishably)
  - A warp executes one common instruction at a time (efficiency requires that all threads in the warps do the same thing, take the same branches, do the same operation).
  - If multiple execution path are possible (due to branches), if the decision on which branch to take is not complete between the threads all of the possible execution path are executed!
    - This also hint that atomic operations issued by threads in a warp to the same memory location would be executed sequentially
  - occupancy
- More info @ https://docs.nvidia.com/cuda/cuda-c-programming-guide/

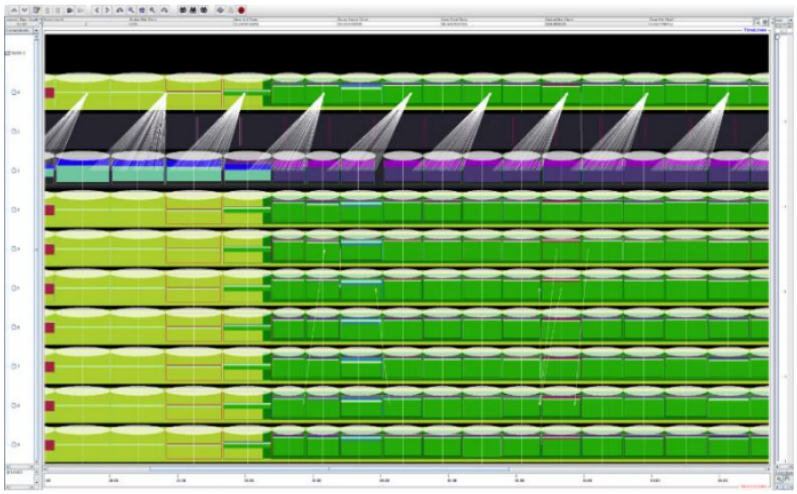
Debugging

- Commercial tools (DDT, TV, …)
- If possibility to export xterm:
  mpirun –np 2 xterm –e gdb –args <my app args>
- If not, add a sleep (or a loop around a sleep in your applications) and use ”gdb –p <pid>” to attach to your process (once connected to the same node where the application is running)
- gdb can execute GDB commands from a FILE (with --command=FILE, -x)

Profiling

- Non-CUDA application: valgrind (free), or vtune (Intel), Score-P, Tau, Vampir
- CUDA application: nvprof from CUDA





Possible code optimizations

- CUDA:
  - As the computation is symmetrical and highly balanced, one can have a different work distribution and do more computations per thread
  - Use shared memory
  - Divide the computations in 2 parts: what needs external data and what doesn't.
- MPI:
  - Use datatypes
  - Use RMA
- Overlap communication and computations
  - Create a specialized kernel to pack and unpack all the borders in one operation
  - As starting a kernel has a high latency merge this pack/unpack kernel with the updates based on the ghost regions