GPU support for implicit modeling

Juraj Starinsky

Overview

- Implicit models and modeling
- Evaluating implicit function f
- General parallel and GPU processing
- Evaluating implicit function f on GPU
- GPU integration
- GPU processing
- CPU vs GPU implicit modeling

Implicit models

- Model implicitly defined by function

$$f : \mathbb{R}^3 \rightarrow \mathbb{R}$$
$$\forall P \in \mathbb{R}^3$$
$$f(P) = 0 \quad surface$$
$$f(P) < 0 \quad interior$$

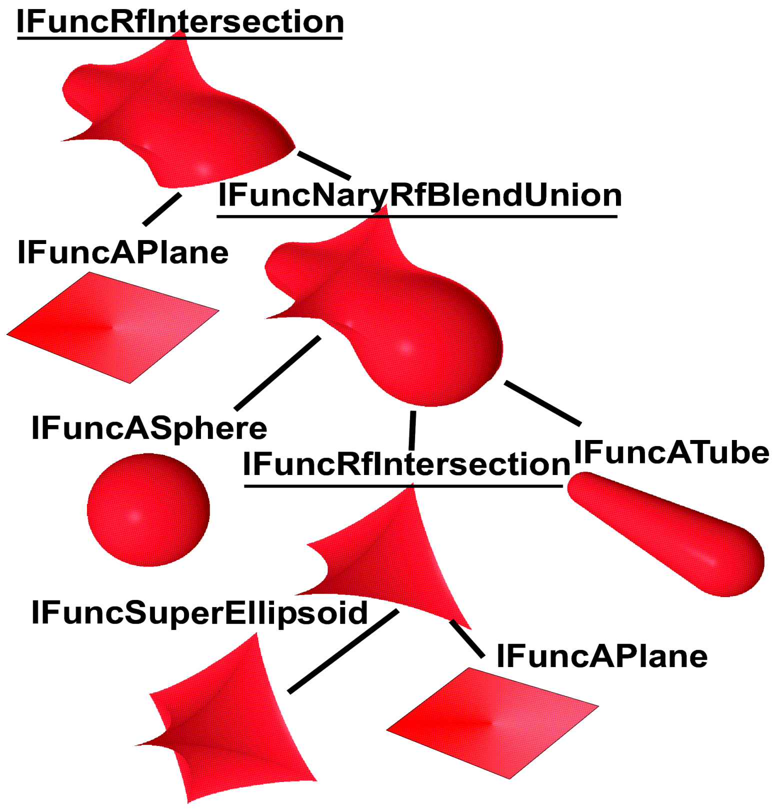
  - Continuous signal
- Complex Models
  - One complex function
    - Hard to control shape

Implicit modeling

- Modeling
  - User interaction – fast response
  - Iterative process – refinement
- Model construction
  - Modeling Tree

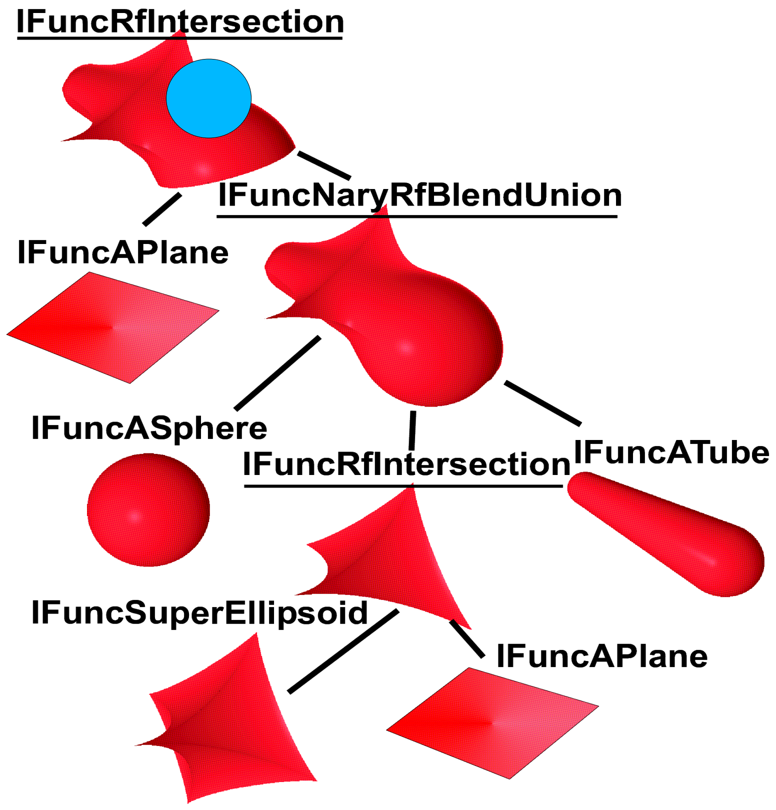
Implicit modeling
modeling tree

- N-ary tree



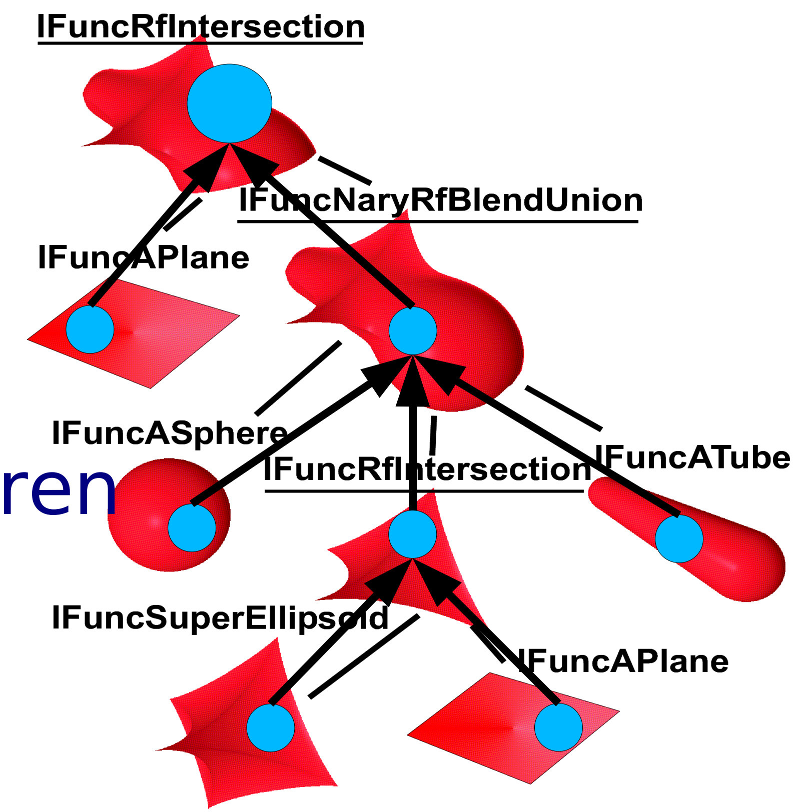
Implicit modeling
modeling tree

- N-ary tree
- Final object in Root



Implicit modeling
modeling tree

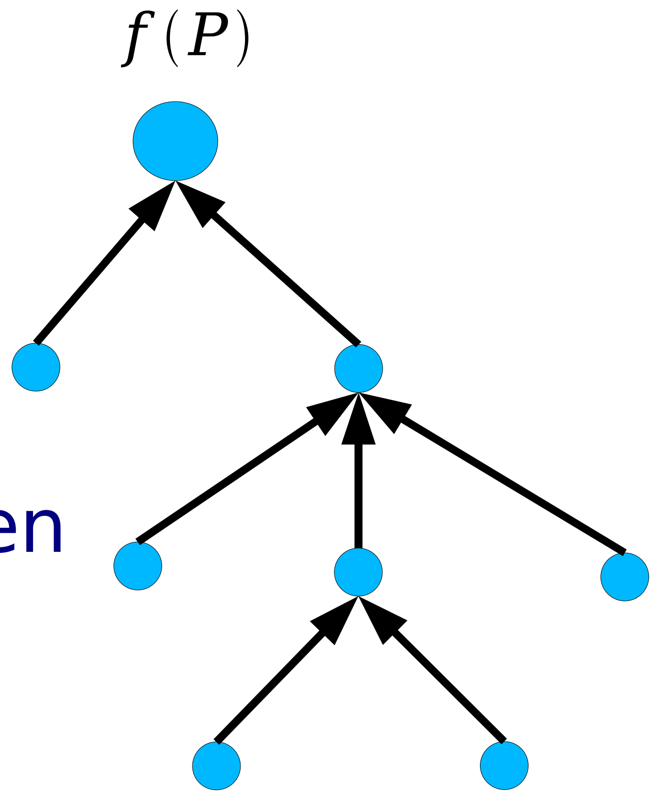
- N-ary tree
- Final object in Root
- Function f(P)
  - Composition of children



Implicit modeling

modeling tree

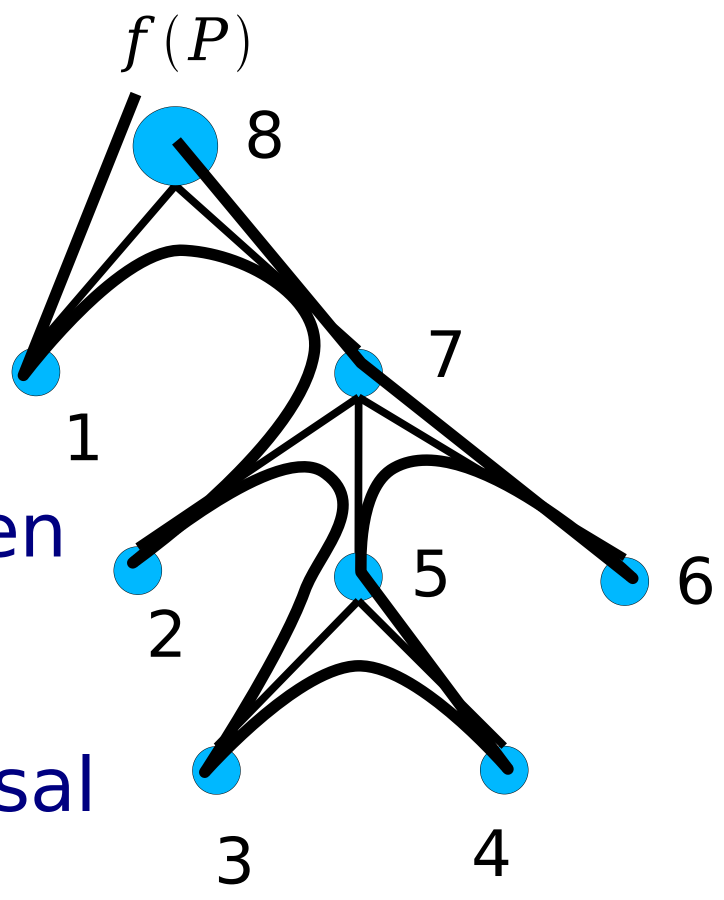
- N-ary tree
- Final object in Root
- Function f(P)
  - Composition of children
- Evaluating f(P)



Implicit modeling

modeling tree

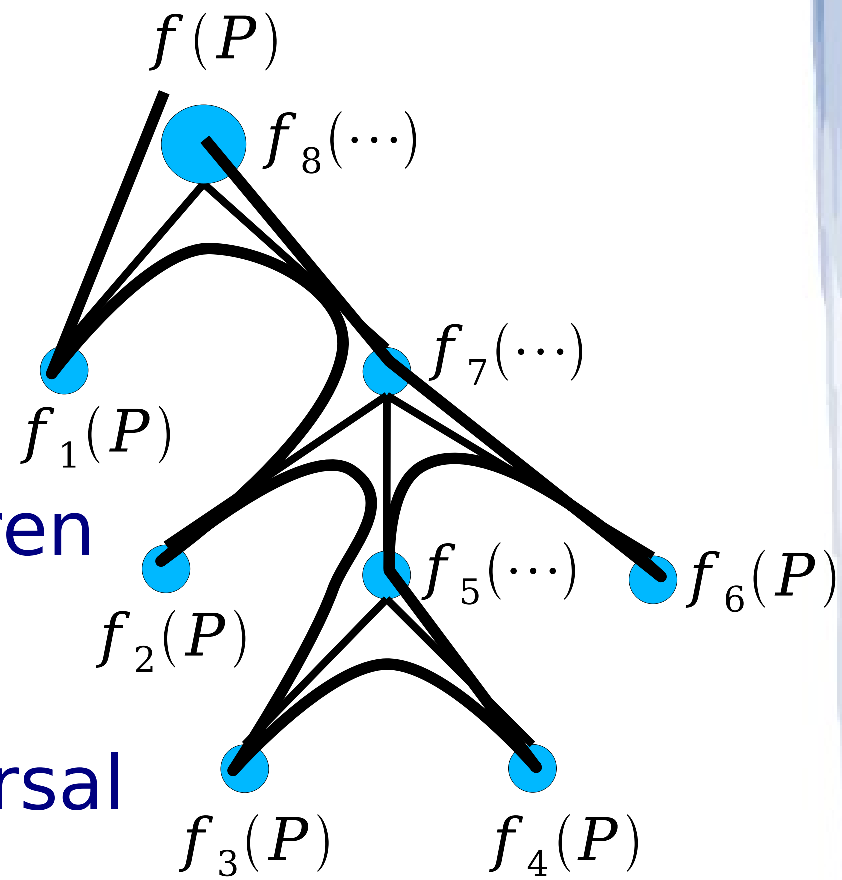
- N-ary tree
- Final object in Root
- Function f(P)
  - Composition of children
- Evaluating f(P)
  - Post-order tree traversal



Implicit modeling

modeling tree

- N-ary tree
- Final object in Root
- Function f(P)
  - Composition of children
- Evaluating f(P)
  - Post-order tree traversal
  - Evaluate node f_i using children's results
- Like CSG



Evaluating f(P)

- Voxelization
  - Volume processing
  - Isosurface extracting
    - Marching cubes/tetrahedra
- Ray-tracing
- Deformation
- ...

Parallel processing

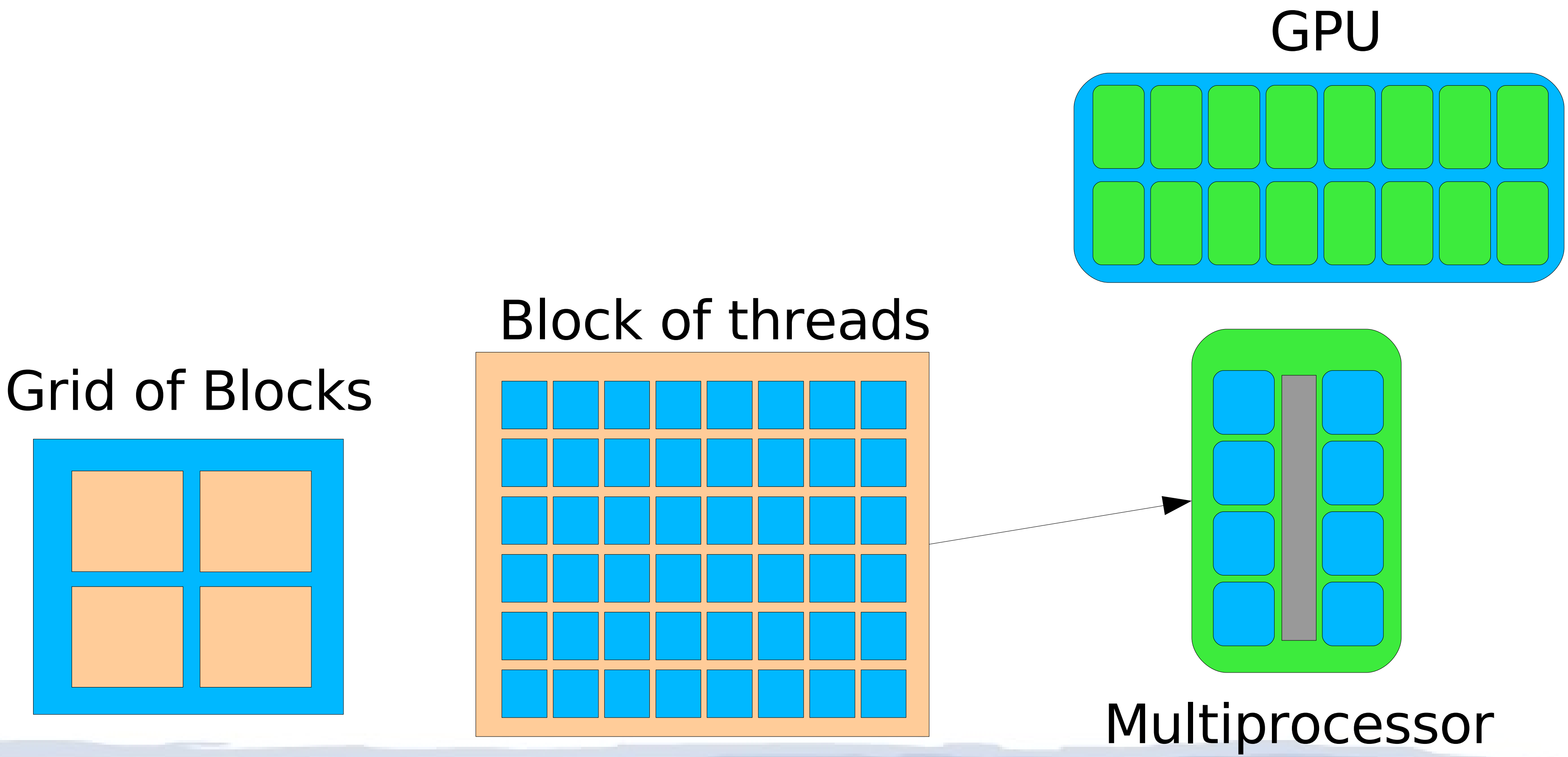
- Task parallelism
  - Different tasks run in parallel
- Data parallelism
  - Same computations operating on different data in parallel
- Instruction parallelism
  - Some instructions within computation can be issued in parallel

GPU processing

- Kernel – program for GPU
- Computation Thread (work-item)
  - Running kernel
- Data parallelism
  - Block of threads (work-group)
    - Grid of threads running the same kernel in parallel sharing some data
  - Grid of blocks
    - Grid of identical Blocks
    - NO sharing/communication between blocks

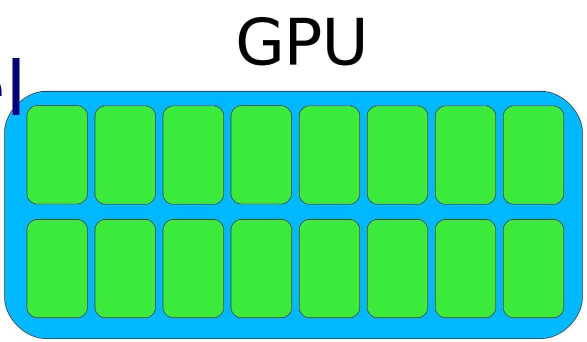
GPU processing

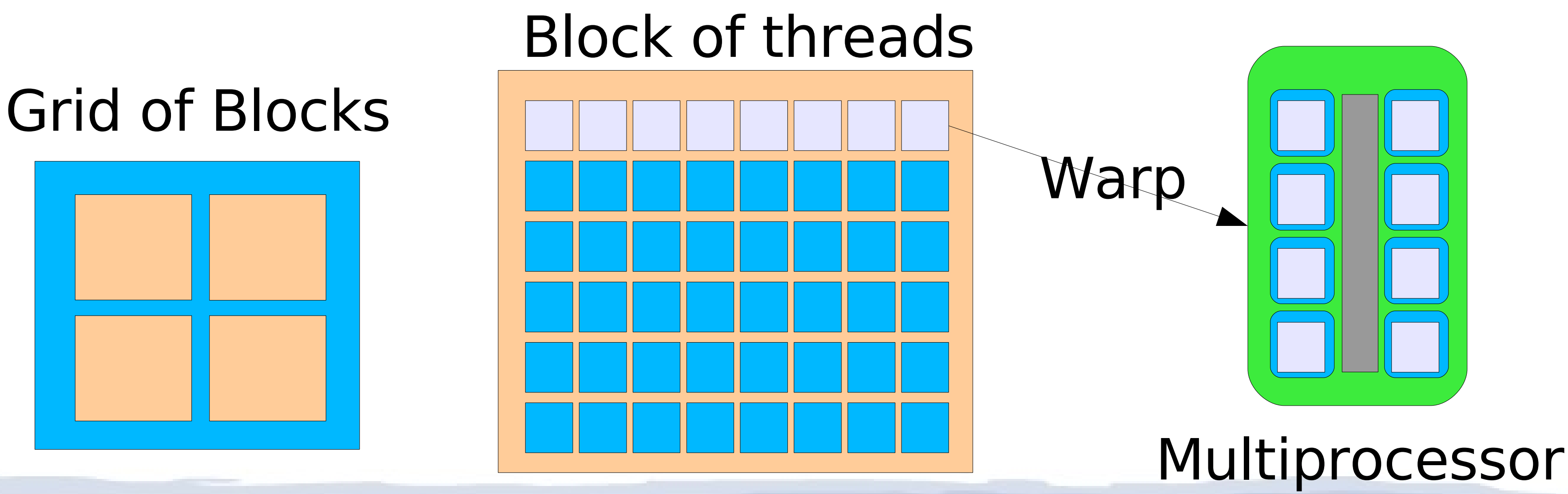
- Execute block on one multiprocessor



GPU processing

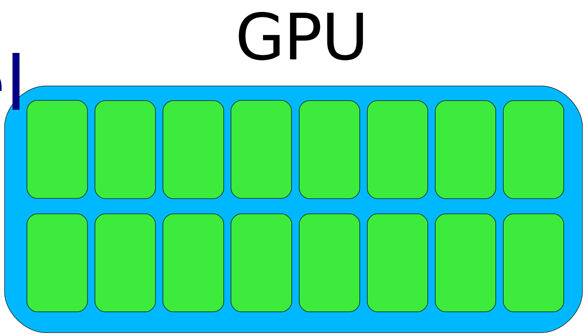
- Execute block on one multiprocessor
  - Divide block into warps
  - Whole warp runs in parallel

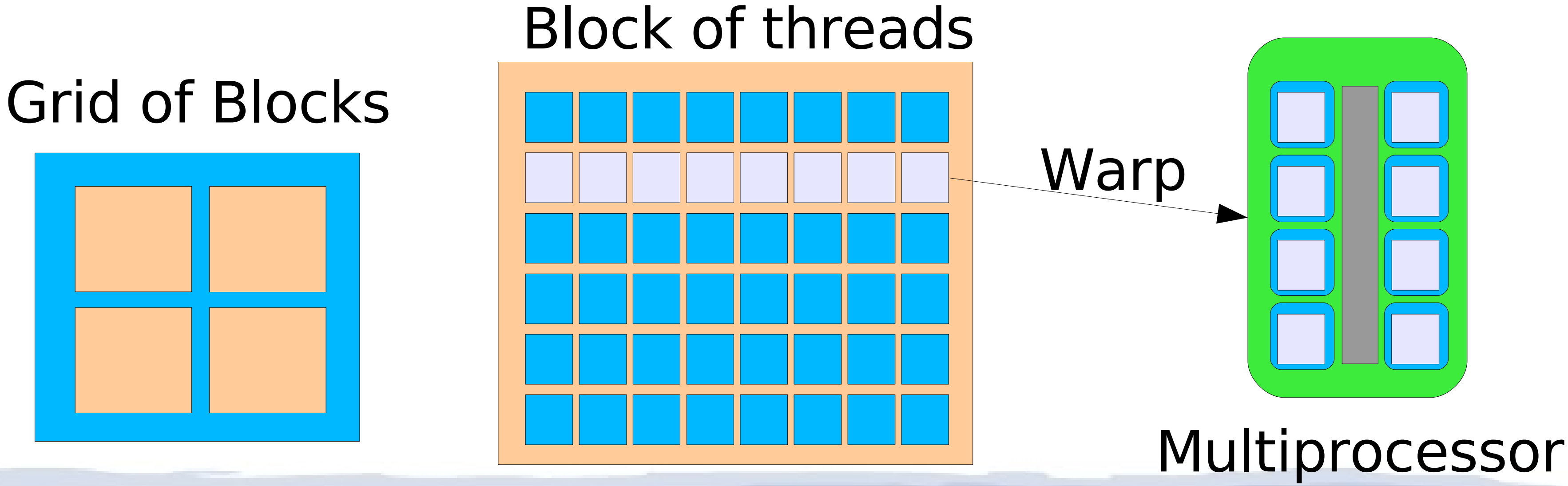




GPU processing

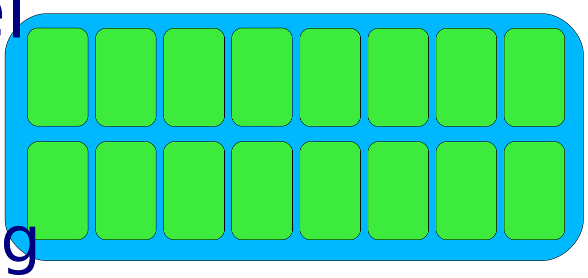
- Execute block on one multiprocessor
  - Divide block into warps
  - Whole warp runs in parallel
  - Switching warps

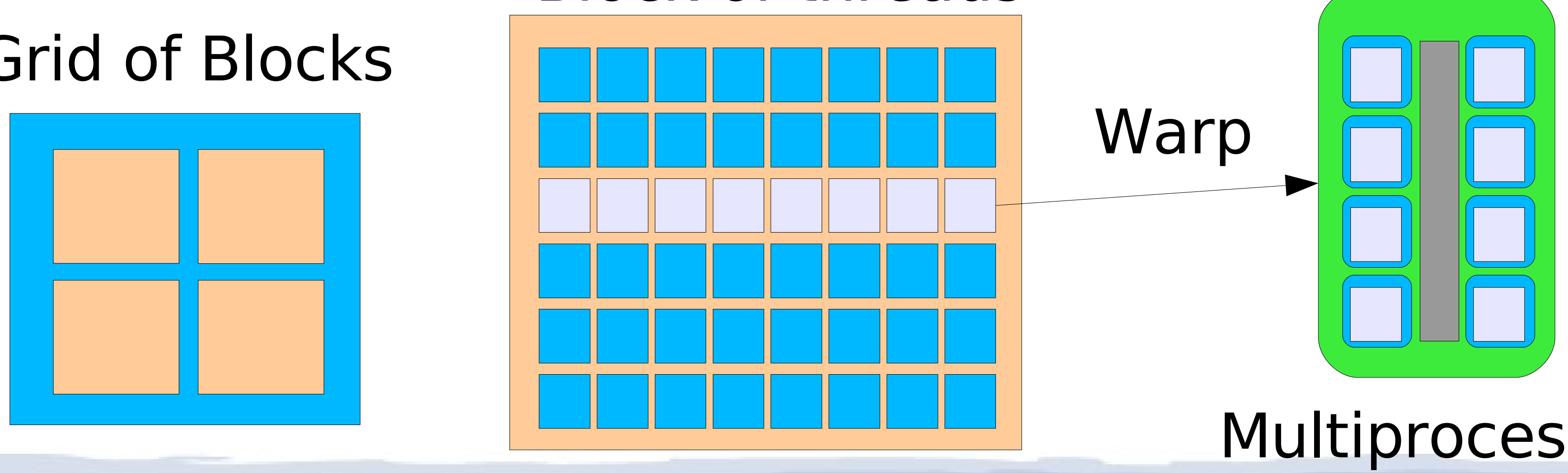




GPU processing

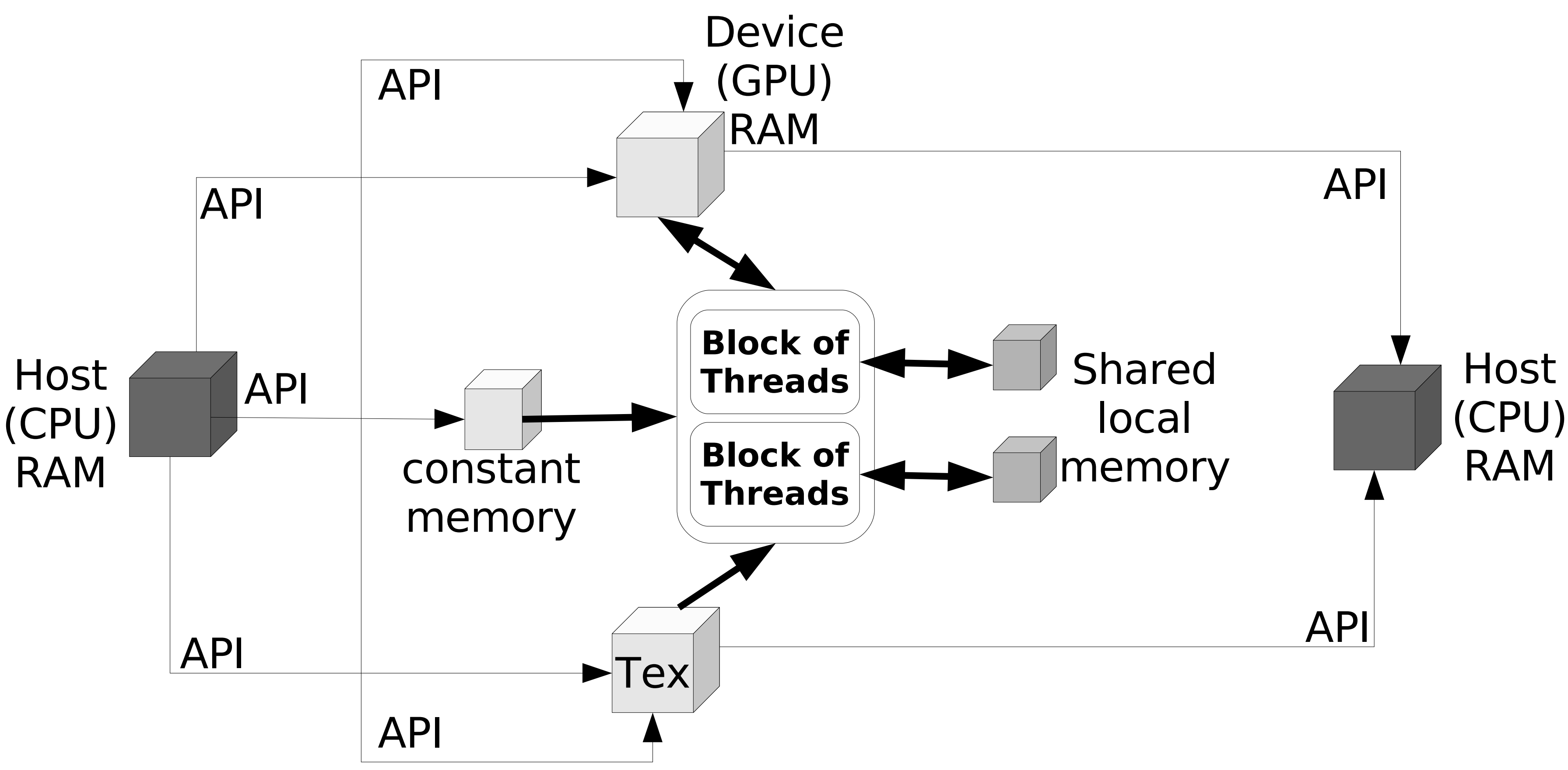
- Execute block on one multiprocessor
  - Divide block into warps
  - Whole warp runs in parallel
  - Switching warps
    - Multiprocessor time slicing





GPU processing

- Data processing



Efficient GPU processing

- Data parallelism
- High arithmetic intensity
  - # of AI per I/O
- Minimum syncs

Evaluating f(P) on GPU

- Evaluate f on a set of points
  - Independent evaluating of points with the same function f
    - Data parallelism, minimum syncs
  - Children's f's are complex functions with some arguments
    - Same constant arguments for all points
    - arithmetic intensive(?)
  - Input = one point P(x,y,z)
  - Output = one value f(P)

Evaluating f(P) on GPU

pitfalls

- Implicit modeling systems are OOP
  - Massive virtual method overloading
  - GPU programming does not support OOP
- Evaluating of N-ary tree is recursive
  - GPU do not support recursion/stack
    - Recursion is expanded
- f(P) can be arbitrary
  - Branching within f
    - GPU is fast if all threads in warp follow the same computation path

GPU integration

into implicit modeling system

- GPU programming does not support OOP
  - Non-OOP f's for GPU
    - Virtual method calls ~ switch statement
      - Lot of branching (but same path)
  - For every loaded n-ary tree, compose the exact f's GPU source code
    - No Switch statement
    - Run-time compiling of GPU source
    - Composed/compiled GPU source can be saved and analyzed/reused

GPU integration
into implicit modeling system

- Evaluating of N-ary tree is recursive
  - Recursion is expanded, no stack
  - intermediate values are stored in local memory – consumes registers
  - Registers are limited
    - Fail to execute, if too many threads in block
    - Exceeding limit will use global (slow) memory – decreasing arithmetic intensity

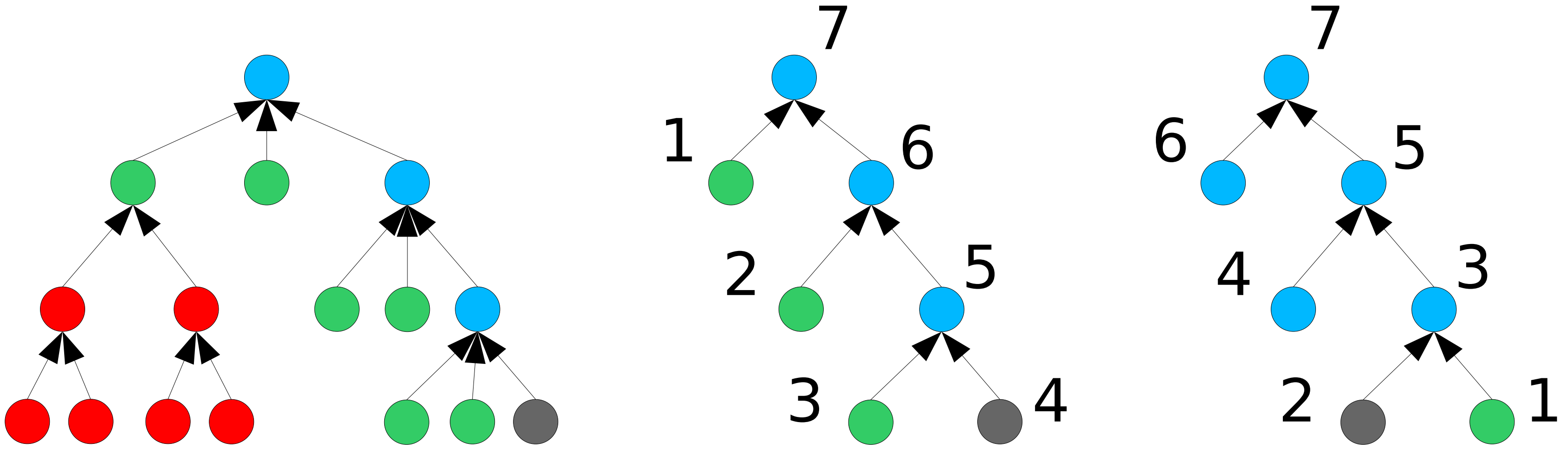
GPU integration

N-ary tree traversal

- Expanded recursion storage size
  - Sub-tree output = 1 value
  - Node requires all values from its subtrees (n)
  - One node is being evaluated at a time
  - Every visited not finished node has at most n-1 values from its subtrees ready
  - There are at most h not finished nodes (path from root to actual visited node in a tree of height h is at most h)
  - Total at most h*(n-1)+1

GPU integration
N-ary tree traversal

- Post-order traversal
  - Left – Right – Middle
  - Right – Left – Middle
  - If (left>right) LRM else RLM



GPU integration

branching within f

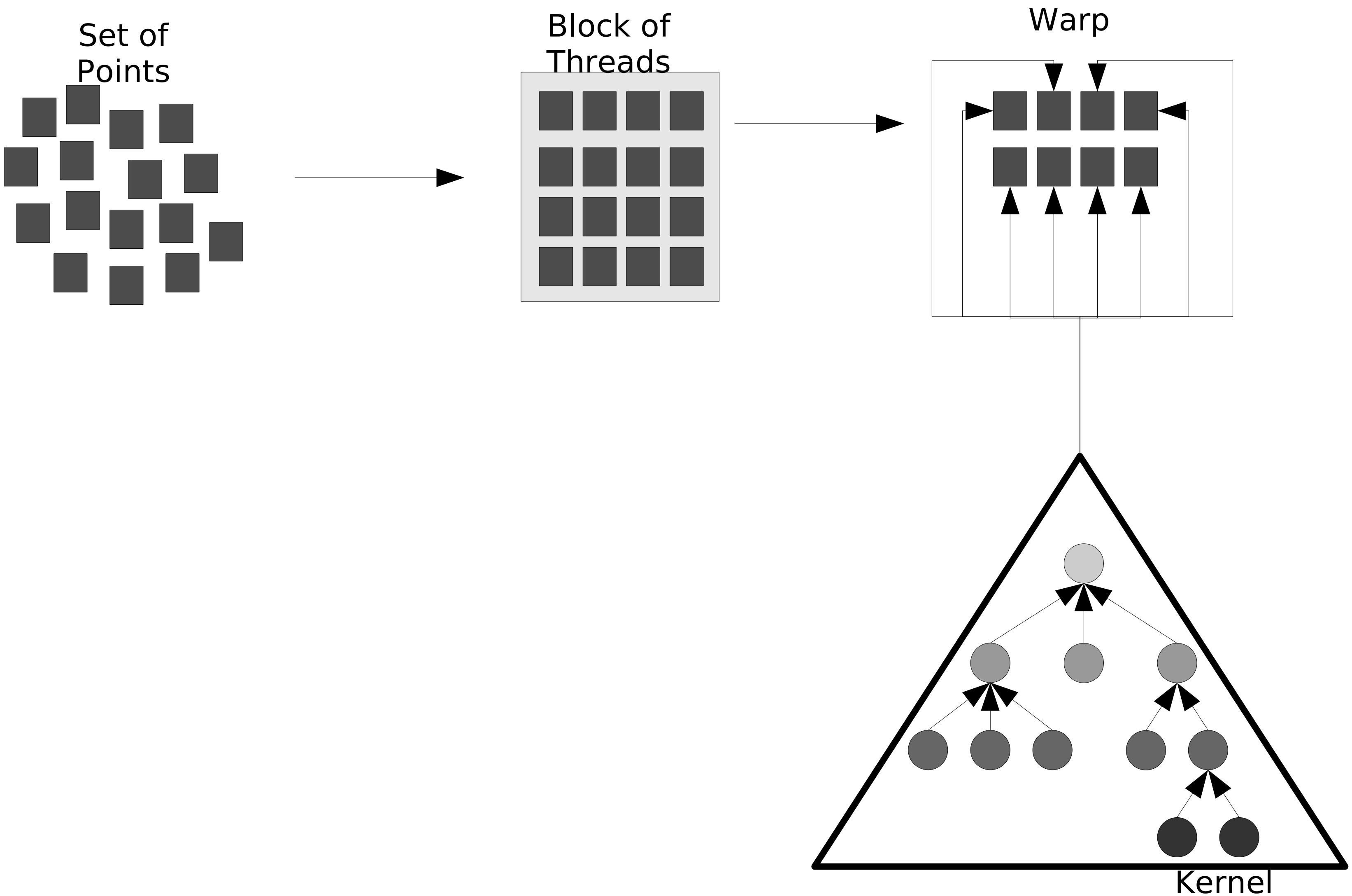
- Divergent branches are serialized
  - Parallel performance decreases to serial
- Spatial coherency
  - Evaluations of points close to each other are assumed to follow the same path

GPU integration
constant arguments of f

- Only one instance of arguments
  - f is same for all points
- Store in constant memory (fast)
  - If it does not fit ?
    - Some f's may have m^2 arguments
    - Texture – better caching then global mem
      - Addressing math and swizzling
    - Shared mem – need copy from global/texture once for every block of threads

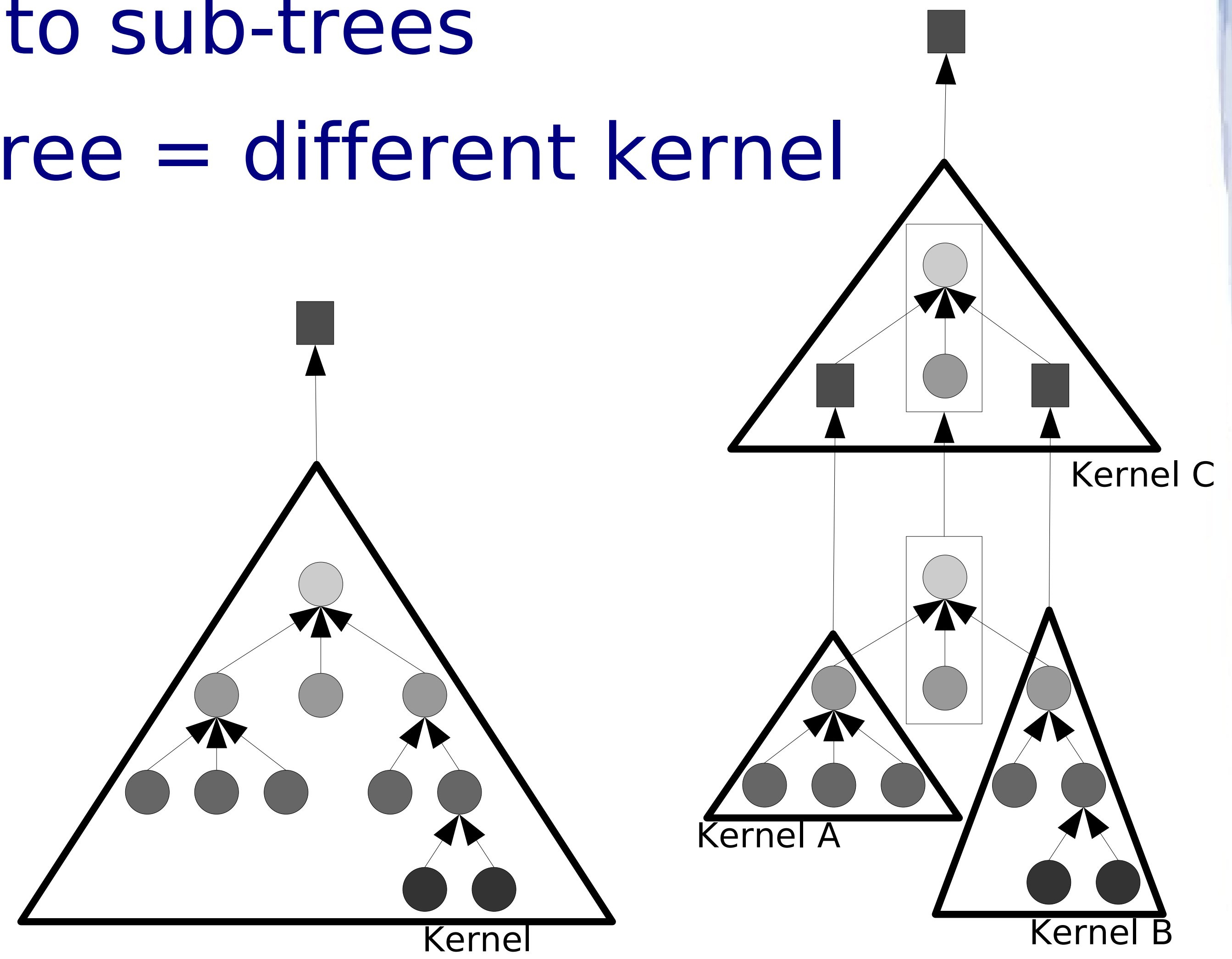
GPU processing

general case



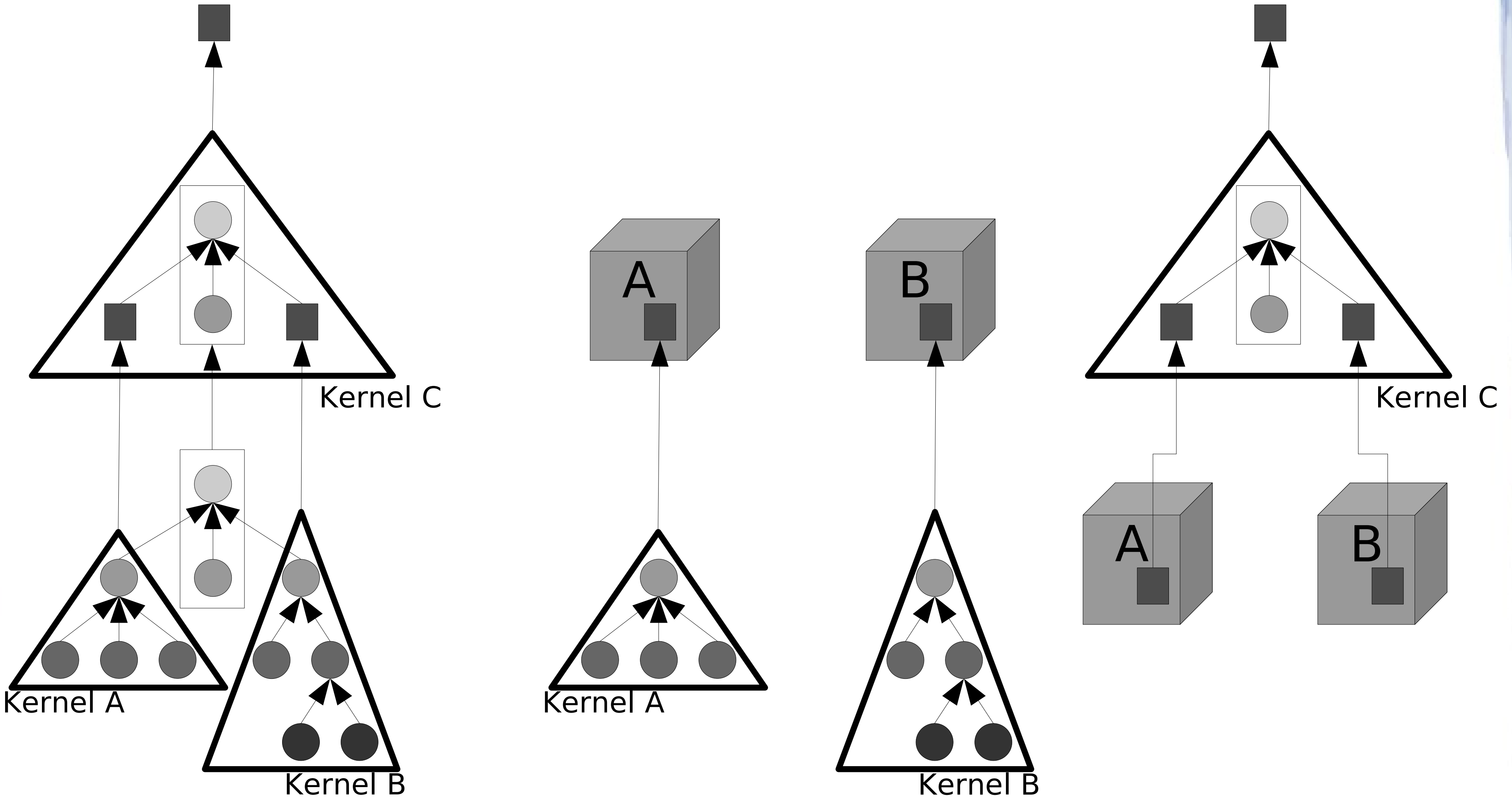
GPU processing
large tree

- Split tree into sub-trees
- Every sub-tree = different kernel



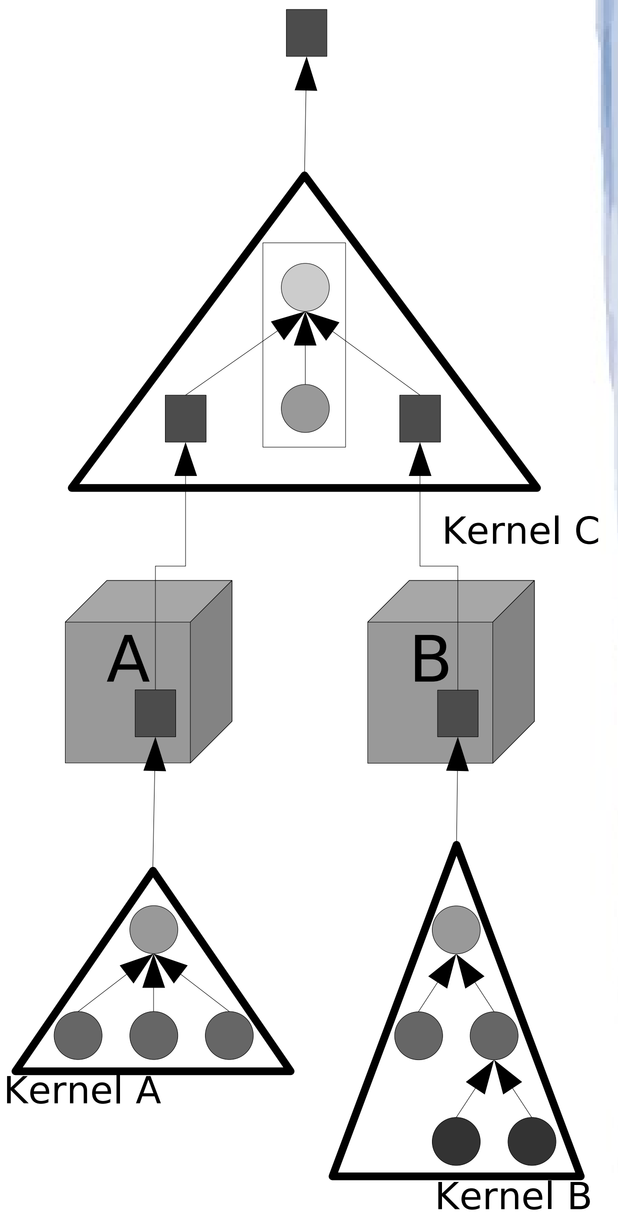
GPU processing

large tree



GPU processing
sub-trees

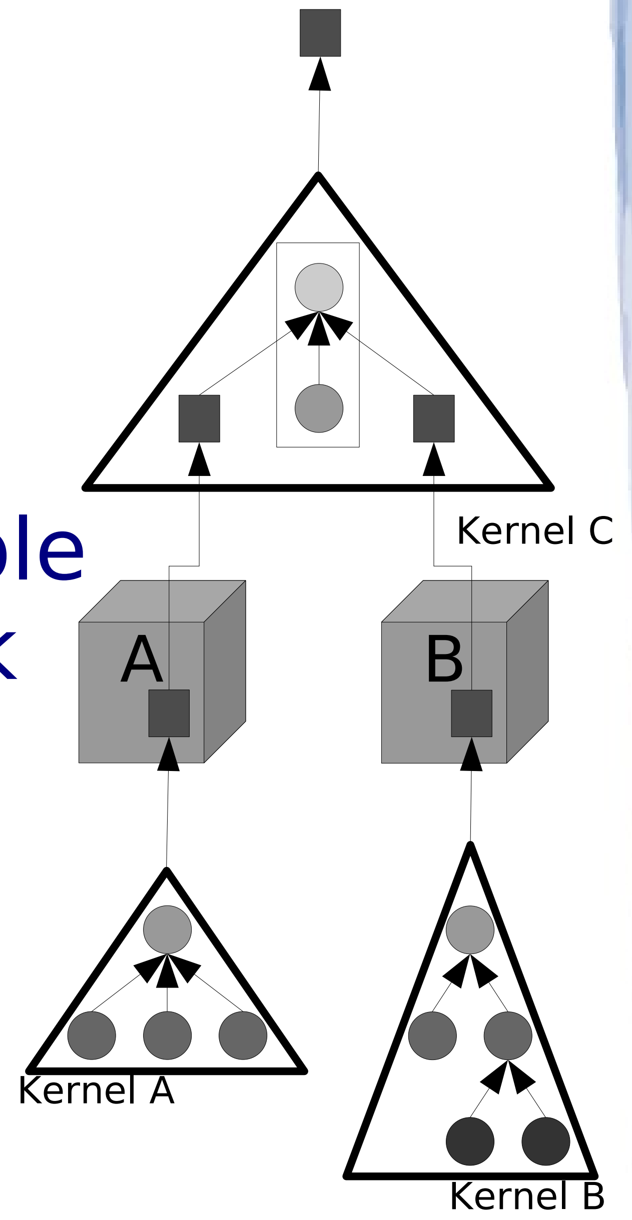
- Switching kernels
- Buffers for intermediate values
  - Buffer size ~ # of kernel switches
- Kernels on the same level can be processed in parallel
  - Task parallelism
    - Computation streams



GPU processing

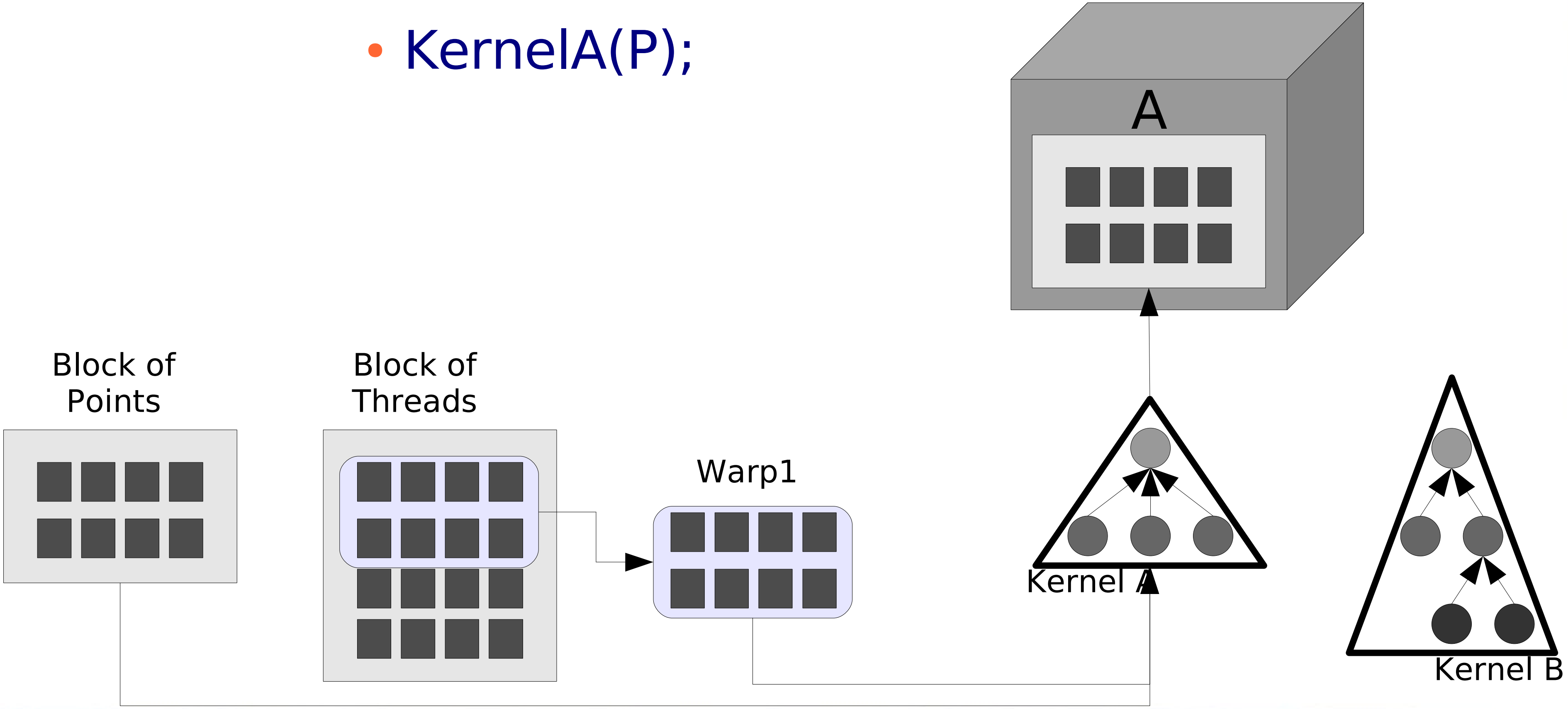
sub-trees

- Large tree but few points
  - No utilization for large number of threads ?
- Parallel processing of multiple subtrees within thread block
  - Exploit warps !



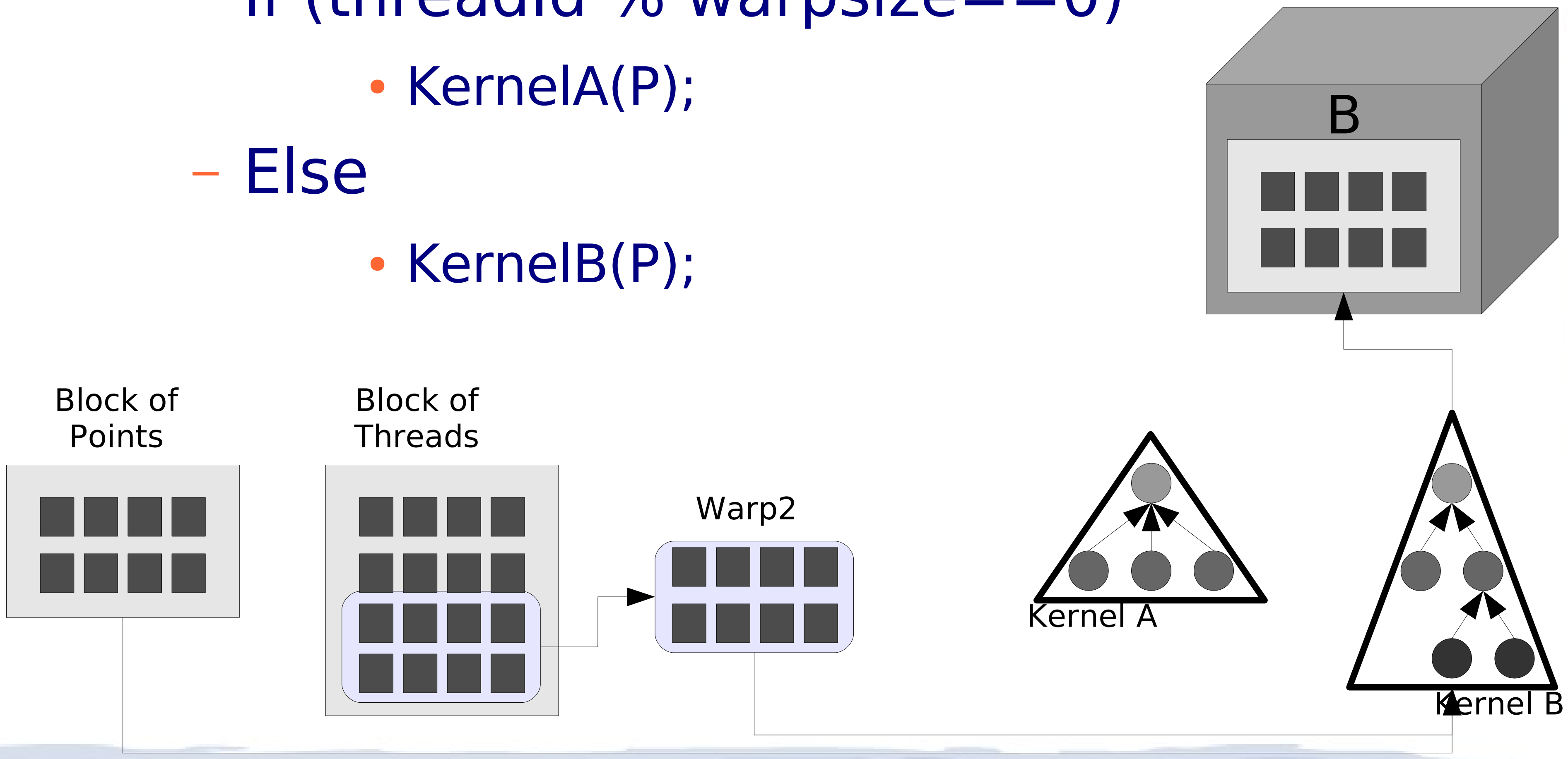
GPU processing
sub-trees in thread block

- Kernel
  - If (threadId%warpsize==0)
    - KernelA(P);



GPU processing
sub-trees in thread block

- Kernel
  - If (threadId % warpsize==0)
    - KernelA(P);
  - Else
    - KernelB(P);



GPU vs CPU implicit modeling

- Interactive modeling
  - interactive updating
  - interactive visualization
    - Volume data
    - Mesh data
    - Ray-tracing

GPU vs CPU implicit modeling

- Interactive modeling
  - interactive updating
  - interactive visualization (mesh data)
- CPU
  - Updating - CPU producing triangulation
    - Marching cubes on CPU
  - Visualization – GPU requires triangulation
    - Transfer triangulation from CPU to GPU for every update - slow

GPU vs CPU implicit modeling

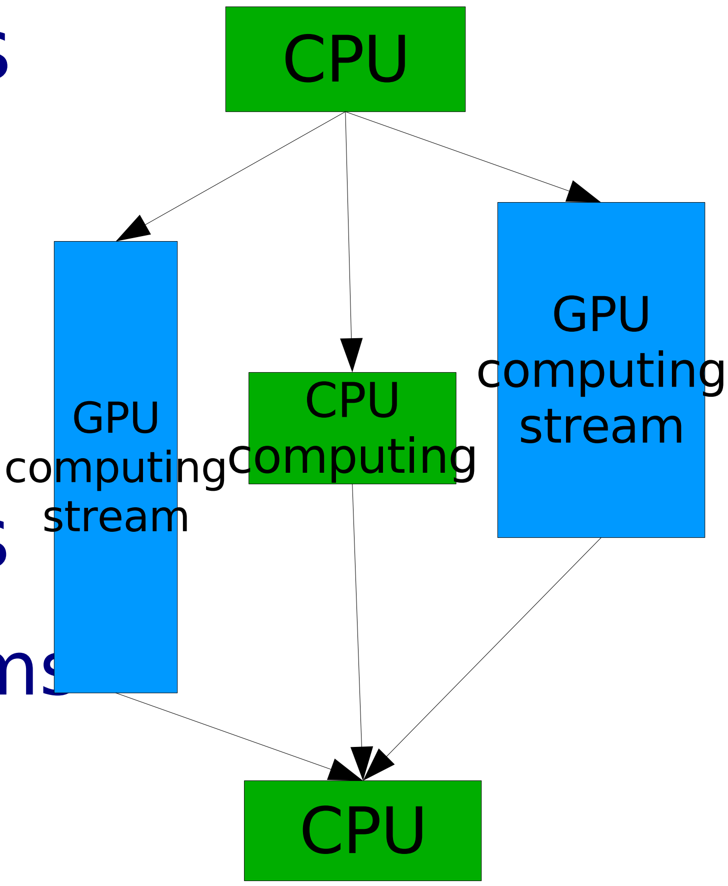
- Interactive modeling
  - interactive updating
  - interactive visualization (mesh data)
- GPU
  - Updating - GPU producing triangulation
    - Marching cubes on GPU
  - Visualization – GPU requires triangulation
    - Everything is done on GPU – no slow transfer needed

GPU vs CPU implicit modeling

- GPU raytracing implicit models
  - Interactive framerate ?
    - for camera movement use triangulated data
    - for static camera - raytrace

Multiple parallel computations

- asynchronous operations
  - copy CPU <-> GPU
  - GPU processing
  - CPU is free to work
- parallel async operations
  - multiple computing streams
    - own copying & processing



References

- NVIDIA CUDA Programming Guide 1.0, 1.1, 2.0, 2.1
- The CUDA Compiler Driver NVCC
- Nvidia Geforce GTX 200 GPU architecture overview
- AMD Stream computing user guide
- AMD Entering the golden age of Heterogeneous Computing
- PODLOZHNYUK V.: Image convolution with CUDA