Genetic Algorithms on GPUs

Sungjoo Ha
shurain@soar.snu.ac.kr

Introduction

- Using GPUs for general purpose computations is becoming increasingly popular
  - High performance
  - Low cost
  - Ubiquitous availability
- Genetic algorithm exhibits nice properties which promotes the use of parallel computing platforms
  - Each individual is independent of each other in evaluation of fitness value
  - Operators usually involve one or two individuals

GPU Execution & Memory Model

- Execution model
  - Function/task run on GPU is called *kernel*
  - Each kernel is distributed among *blocks* which makes up a *grid*
  - Each block bundles a group of *threads*
- Memory model
  - GPU have a *global memory* which is used to transfer data between host and device
  - Threads in a block may communicate with each other using *shared memory*
  - Blocks cannot communicate with each other directly

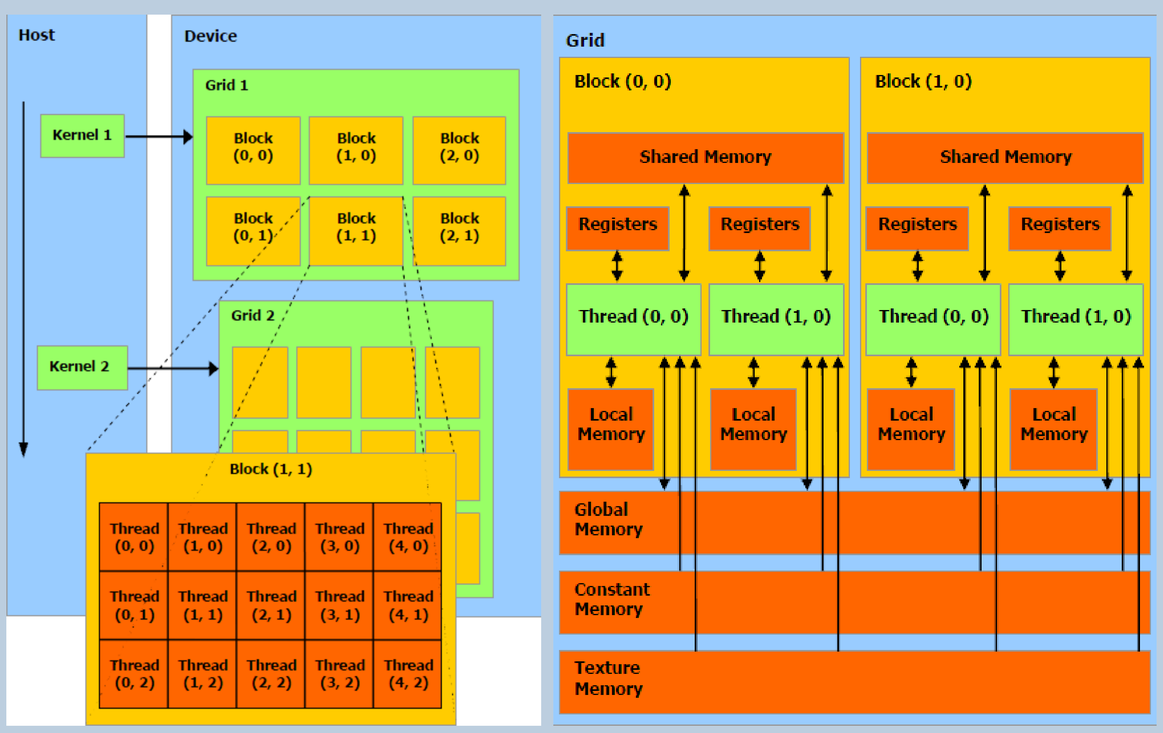

Figure 1: CUDA Execution & Memory Model

Genetic Algorithm

- Inspired by natural selection and evolution
- Effective meta-heuristic for solving NP-hard problems

```
Create random population;
Evaluate fitness;
repeat
    Select parents;
    Crossover parents to obtain child;
    Mutate child;
    Evaluate fitness of child;
    Replace child with an individual from population;
until Some condition;
```

Algorithm 1: Pseudo-code of GA

Master-slave GA

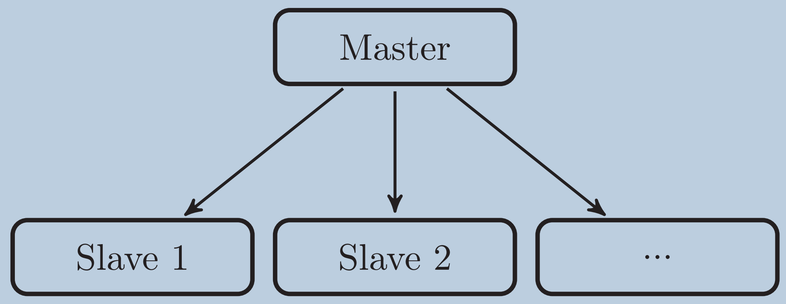

Figure 2: Master-Slave Model

- Single population
- Distribute fitness evaluation to slaves
- Possibly distribute genetic operators as well

Coarse-grained GA

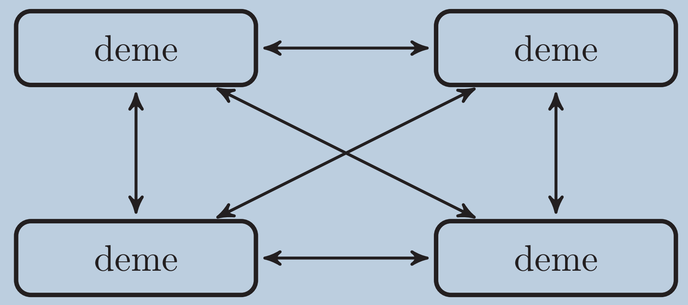

Figure 3: Island Model

Figure 4: Stepping-Stone Model

- Multiple subpopulations evolve in parallel
- Migration exchanges individuals in different subpopulations

Fine-grained GA

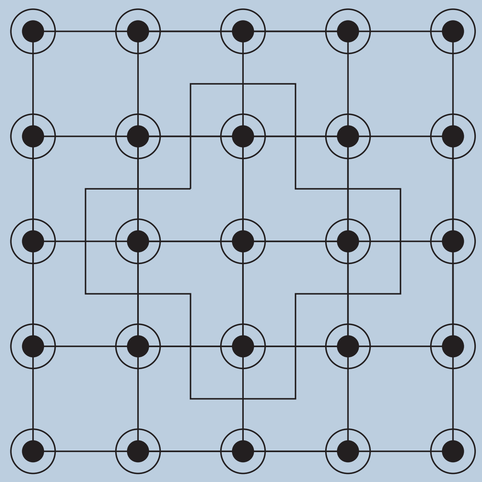

Figure 5: Toroidal Grid Model

- Single population
- Individuals limited to only interact with its neighbors

Challenges

- Careful planned memory access pattern
  - Squeezing every necessary value into shared memory
  - Concise representation of population
  - Host/device memory tranfer pattern
  - Exploiting parallel program patterns
- Fine tuning using device level knowledge
  - Coalescing memory access
  - Avoid bank conflict