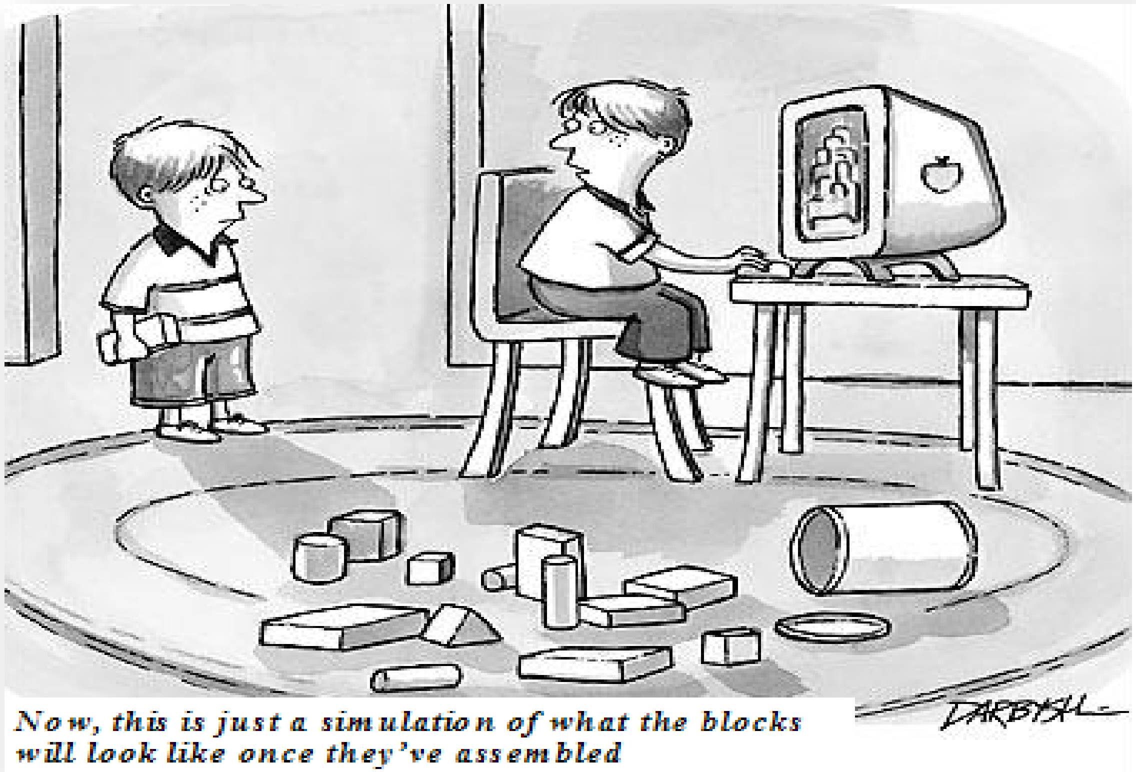
ECE569
Module 4



- GPU Programming Approach

News

- **HPC accounts**
  - Email me after your request
- **HW1**
  - Setting up the environment for programming assignments
- **HW2 coming soon**

THE UNIVERSITY OF ARIZONA

TCRβ Repertoire Modeling Using a GPU-Based In-Silico DNA Recombination Algorithm

Gregory Striemer

Elnaz T. Yazdi

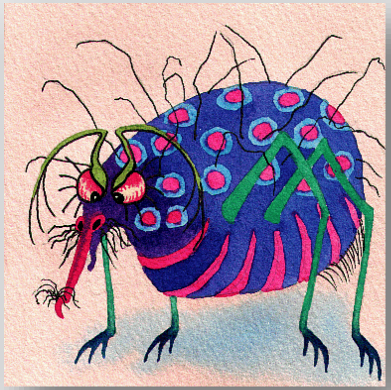
In Collaboration with
Department of Immunobiology
(Jeffrey Frelinger, Adam Buntzman)

DNA Recombination

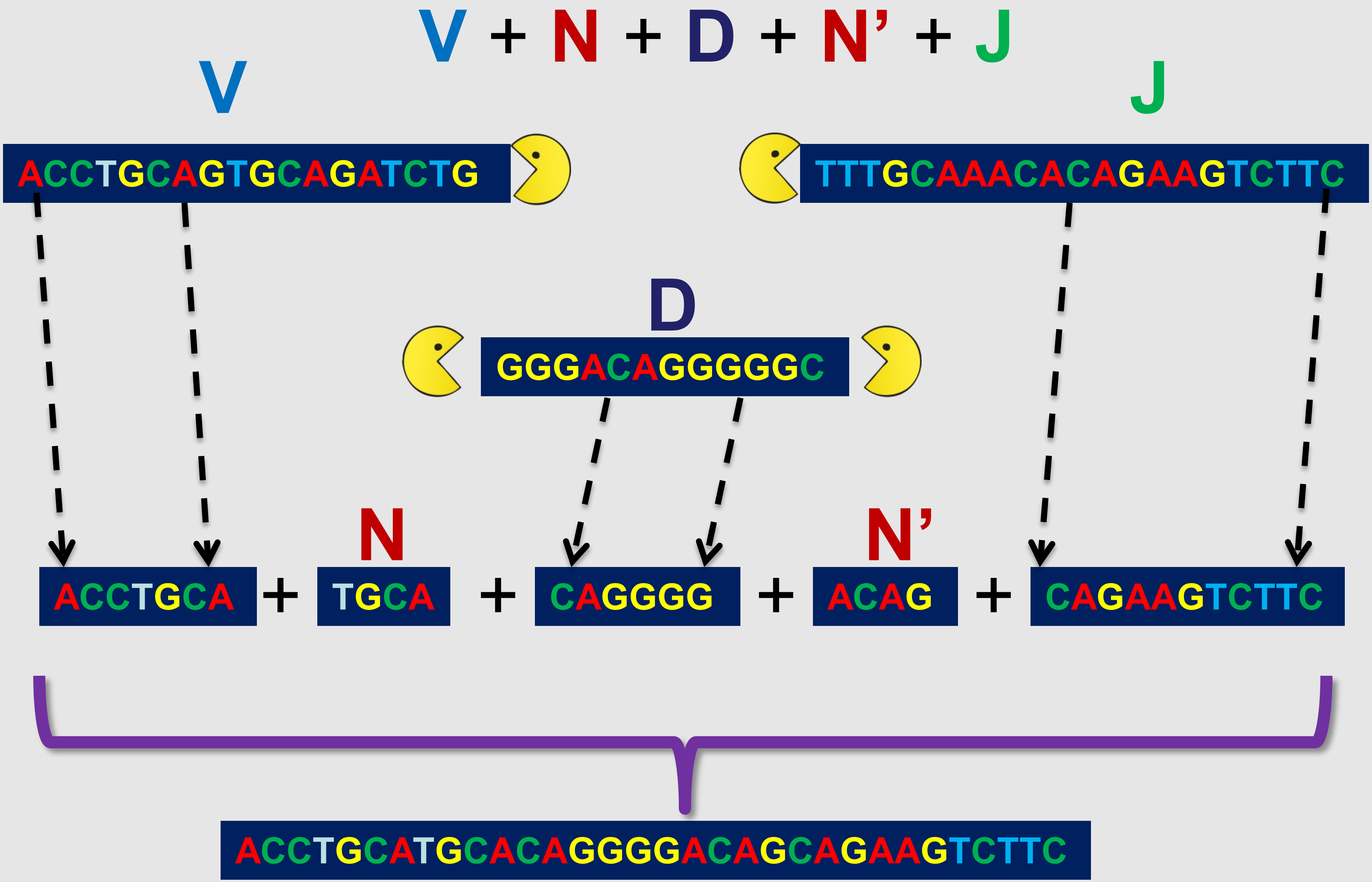
- Immune systems of jawed vertebrates depend on DNA (VDJ) recombination
  - Determines diversity of antigen receptors; Immunoglobulins, T-cell receptors (TCRs)
  - Rearrangement of gene segments to allow for antigen recognition
- Help scientists better understand immune system responses to foreign antigens

DNA Recombination

- Consists of the rearrangement of sets of:
  - Variable (V) Genes (20 sequences)
  - Diversity (D) Genes (2 sequences)
  - Joining (J) Genes (12 sequences)
- Non-templated (n) -Nucleotides create additional diversity at VDJ junctions



DNA Recombination



DNA Recombination

- **V** (362 possible termini variations for 20 seqs)
- **J** (283 possible variations for 12 seqs)
- **D** (230 paths for DB1, 275 paths for DB2)
- **N-Nucleotide**: 4^m (m is length of 'N')
- Possible Recombination Paths with m=10:

  398,292,334,673,920
- Peta-scale Computing
  - Amenable to fine-grain parallelism
- ***Prior studies have only explored sub-sets of recombinant repertoire***

DNA Recombination

In Silico Comparisons		
n_{len}	n_{comb}	Total Comparisons
0	1	13,704,874,784
1	4	109,354,392,096
2	16	655,557,140,224
3	64	3,494,786,848,256
4	256	17,469,380,542,464
5	1,024	83,838,454,767,616
6	4,096	391,197,549,461,504
7	16,384	1,788,165,119,410,176
8	65,536	8,046,160,163,897,344
9	262,144	35,758,639,400,615,936
10	1,048,576	157,330,552,582,569,984

Counting the number of unique pathways, a given sequence can be generated!

DNA Recombination

- Each thread generates a unique n-nucleotide
- Total threads launched is based on length of nucleotide: 4^m threads (always divisible by warp-size when m > 2)
- Each thread performs its own recombination with V, D, and J

GPU Thread and Thread-Block Configuration

N_{len}	$Threads_{Kernel}$	$Threads_{Block}$	Num_{Blocks}
0	1	1	1
1	4	4	1
2	16	16	1
3	64	32	2
4	256	32	8
5	1,024	32	32
6	4,096	64	64
7	16,384	256	64
8	65,536	128	512
9	262,144	128	2,048
10	1,048,576	128	8,192

NVIDIA K20X using CUDA versus Serial Code running on an Xeon 2.83GHz.

2.34 days vs 260 weeks

Goals

- **Current State:**
  - Single GPU (published)
  - Multi-GPU (published)
- **Current Issues**
  - Scalability
  - Memory optimization

GPU

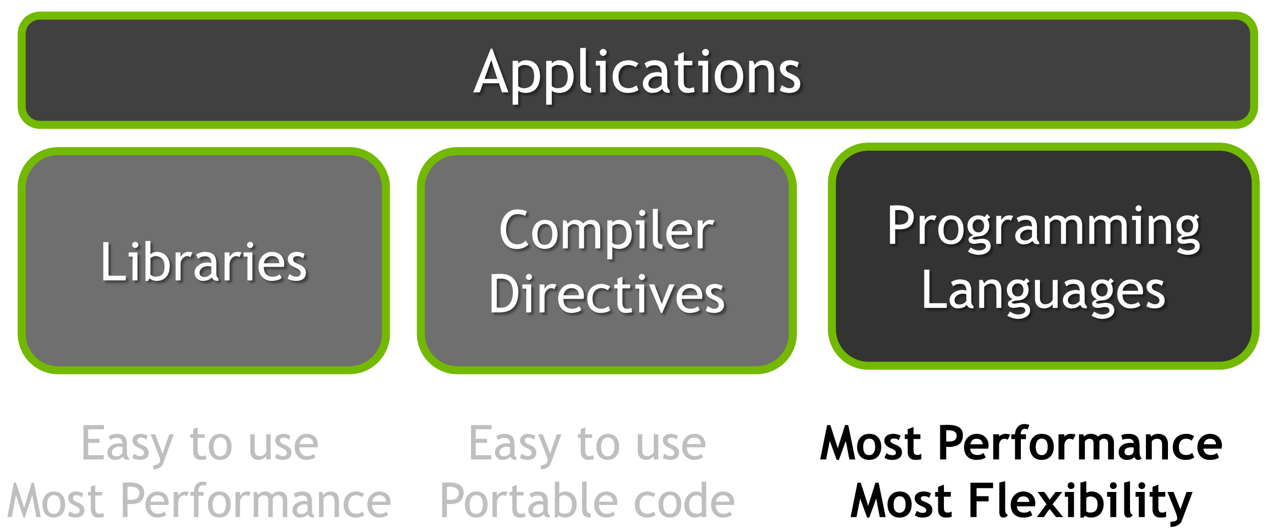
- Basic idea:
  - Heterogeneous execution model
    - CPU is the host, GPU is the device
  - Develop a C-like programming language for GPU
  - Unify all forms of GPU parallelism as CUDA thread
  - Programming model: “Single Instruction Multiple Thread”

CUDA

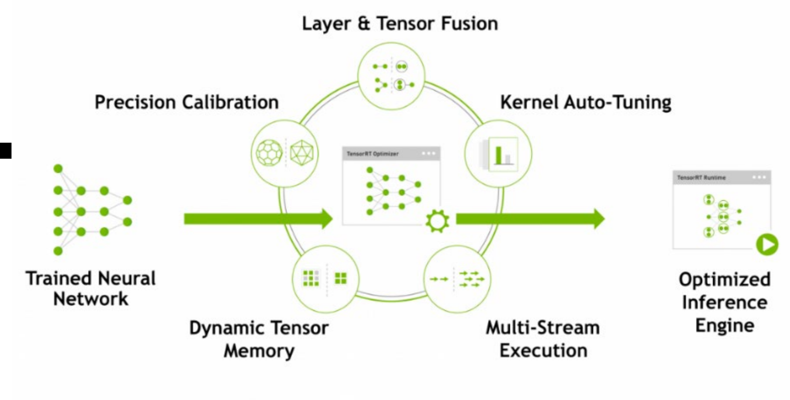
- "Compute Unified Device Architecture"
  - When introduced it eliminated the constraints associated with GPGPU
  - It enables a general purpose programming model
    - User kicks off batches of threads on the GPU to execute a function (kernel)
- Enables explicit GPU memory management
  - CUDA APIs

Programming Approaches

- **Developer resources for GPU computing**
  – At least three ways to accelerate applications
    - Libraries: developed by experts, common routines
    - Compiler Directives: annotated code for compiler as hints
    - **Programming Languages**: Significant coding effort
      – Once you understand the computation on GPU through programming you can easily take advantage of the first two approaches



GPU Accelerated Libraries

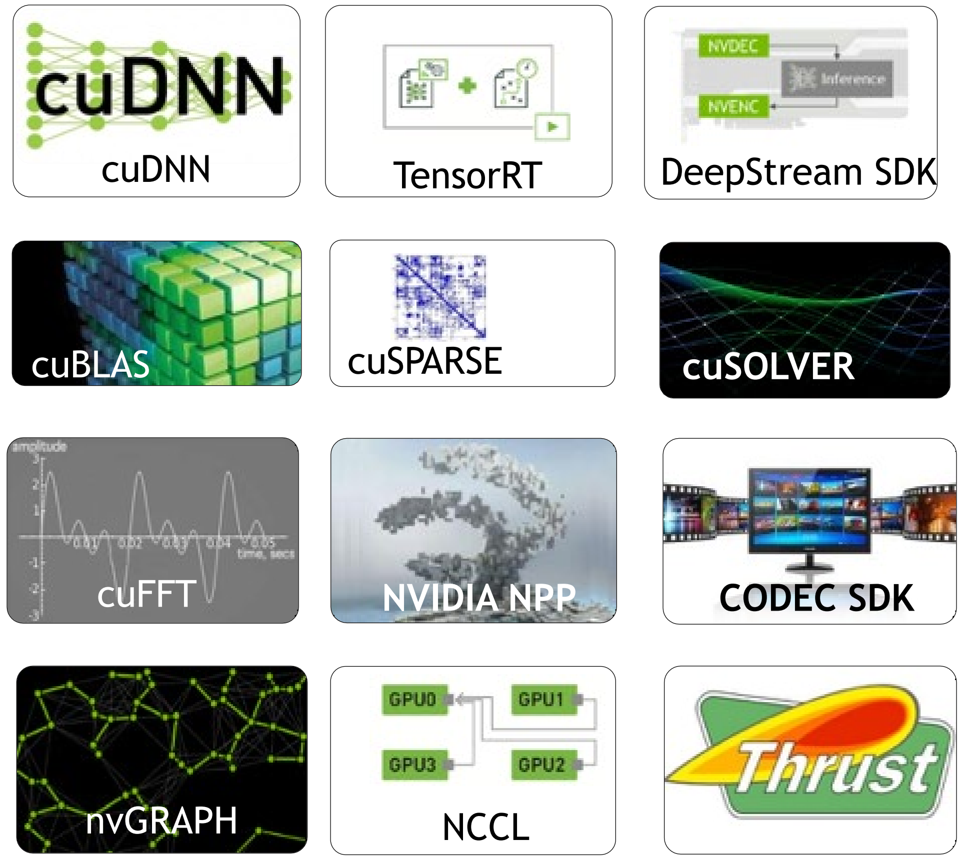


DEEP LEARNING

LINEAR ALGEBRA

SIGNAL, IMAGE, VIDEO

PARALLEL ALGORITHMS



https://developer.nvidia.com/gpu-accelerated-libraries

Libraries

- **When the computation can be partitioned into common patterns**
  – Ease of use: Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
  – “Drop-in”: Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes
  – Quality: Libraries offer high-quality implementations of functions encountered in a broad range of applications

Vector Addition in Thrust Library

```
#include <thrust/device_vector.h>
#include <thrust/copy.h>

int main(void) {
  size_t inputLength = 500;
  thrust::host_vector<float> hostInput1(inputLength);
  thrust::host_vector<float> hostInput2(inputLength);

  thrust::device_vector<float> deviceInput1(inputLength);
  thrust::device_vector<float> deviceInput2(inputLength);

  thrust::device_vector<float> deviceOutput(inputLength);

  thrust::copy(hostInput1.begin(), hostInput1.end(), deviceInput1.begin());

  thrust::copy(hostInput2.begin(), hostInput2.end(), deviceInput2.begin());

  thrust::transform(deviceInput1.begin(), deviceInput1.end(),
          deviceInput2.begin(), deviceOutput.begin(), thrust::plus<float>());
}
```

OpenACC

- **Compiler directives for C, C++, and FORTRAN**

```
#pragma acc parallel loop
copyin(input1[0:inputLength],input2[0:
inputLength]),
copyout(output[0:inputLength])
for(i = 0; i < inputLength; ++i)
{
   output[i] = input1[i] + input2[i];
}
```

Compiler Directives: Easy, Portable Acceleration

- **Ease of use:**
  - Compiler takes care of details of parallelism management and data movement
- **Portable:**
  - The code is generic, not specific to any type of hardware and can be deployed into multiple languages
- **Uncertain:**
  - Performance of code can vary across compiler versions

GPU Programming Languages

Numerical analytics	MATLAB, Mathematica, LabVIEW
Python	PyCUDA, Numba
Fortran	CUDA Fortran, OpenACC
C	CUDA C, OpenACC
C++	CUDA C++, Thrust
C#	Hybridizer

We will learn parallel programming and GPU specific optimization using CUDA C

Programming Languages: Most Performance and Flexible Acceleration

- **Performance:**
  - Programmer has best control of parallelism and data movement
- **Flexible:**
  - The computation does not need to fit into a limited set of library patterns or directive types
- **Verbose:**
  - The programmer often needs to express more details
    - Longer code!

Next

- **Data parallelism vs. Task parallelism**