NVIDIA HPC STANDARD LANGUAGE PARALLELISM, C++

MATT STACK

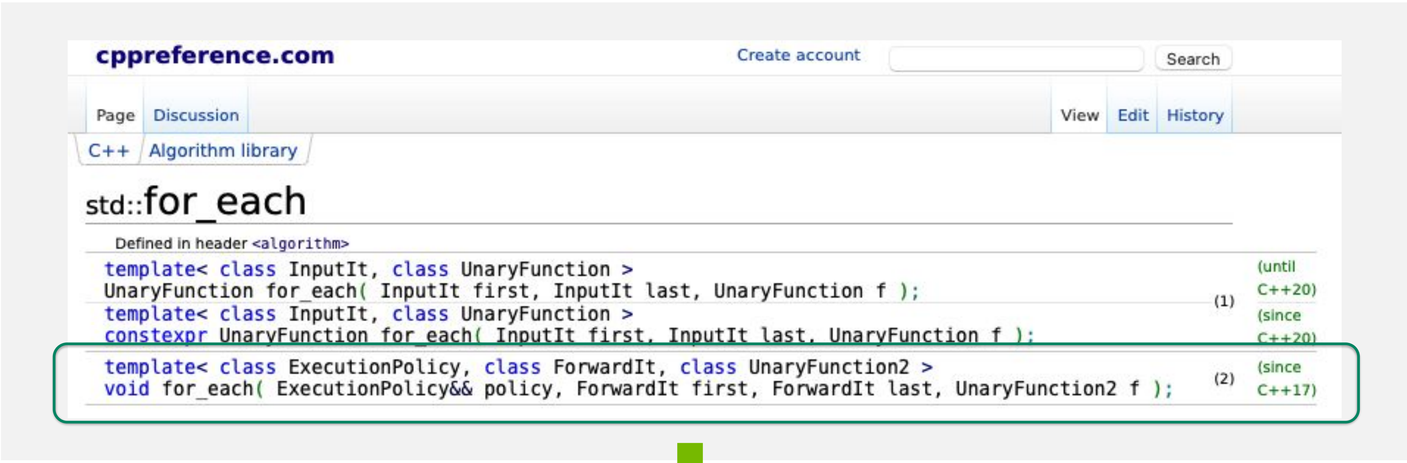
HPC PROGRAMMING IN ISO C++

```
std::sort(std::execution::par, c.begin(), c.end());

std::for_each(std::execution::par, c.begin(), c.end(), func);
```

- Introduced in C++17
- Parallel and vector concurrency via execution policies
  - std::execution::par, std::execution::par_unseq, std::execution::seq
- Several new algorithms in C++17 including
  - std::for_each_n(POLICY, first, size, func)
- Insert std::execution::par as first parameter when calling algorithms
- NVC++ (since 20.7): automatic CPU or GPU acceleration of C++17 parallel algorithms
  - Leverages CUDA Unified Memory

USING C++ STDPAR




```
#include <algorithm> // std::for_each and other functions
#include <execution> // seq, par, par_unseq, un_seq
. . .
std::vector<double> vec = . . .
std::for_each(std::execution::par, vec.begin(), vec.end(), [=](auto i){
        . . . // doing work for each element in the vector
});
```

C++ PARALLEL ALGORITHMS

- When using the parallel execution policy, make sure there are no data races or deadlocks
- StdPar execution on GPU leverages CUDA Unified Memory
  - data needs to reside in heap memory
    - std::vector works
    - std::array does not
- Unlike CUDA C++, functions do not need the __device__ annotation
- Execution on GPU requires random access iterators
- To compile using StdPar, use the –stdpar flag
  - nvc++ -stdpar ./file.cpp
    - -stdpar currently has two options, -stdpar=gpu (which is the default when not given an option) for parallel execution on GPU, and –stdpar=multicore for parallel execution on CPU

C++ PARALLEL ALGORITHMS

Problem: There is a std::vector I want to sort

```
std::vector<int> vec1;
```

{1, 9, 2, 8, 3, 7, 4, 6, 5, 0}

Solution: Using standard algorithm std::sort

```
std::sort(vec1.begin(), vec1.end());
```

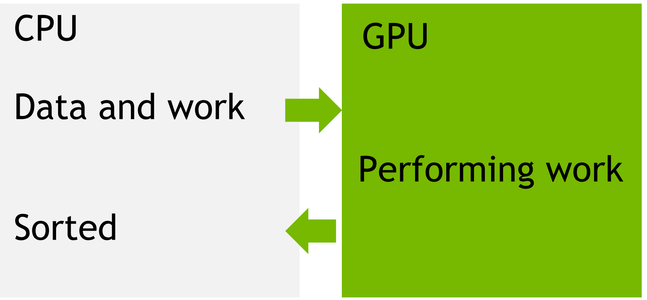
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Potential Performance Improvement: Using parallel execution and –stdpar to offload work and data to GPU

```
std::sort(std::execution::par, vec1.begin(), vec1.end());
```

-during compile-

```
nvc++ -stdpar ./main.cpp
```



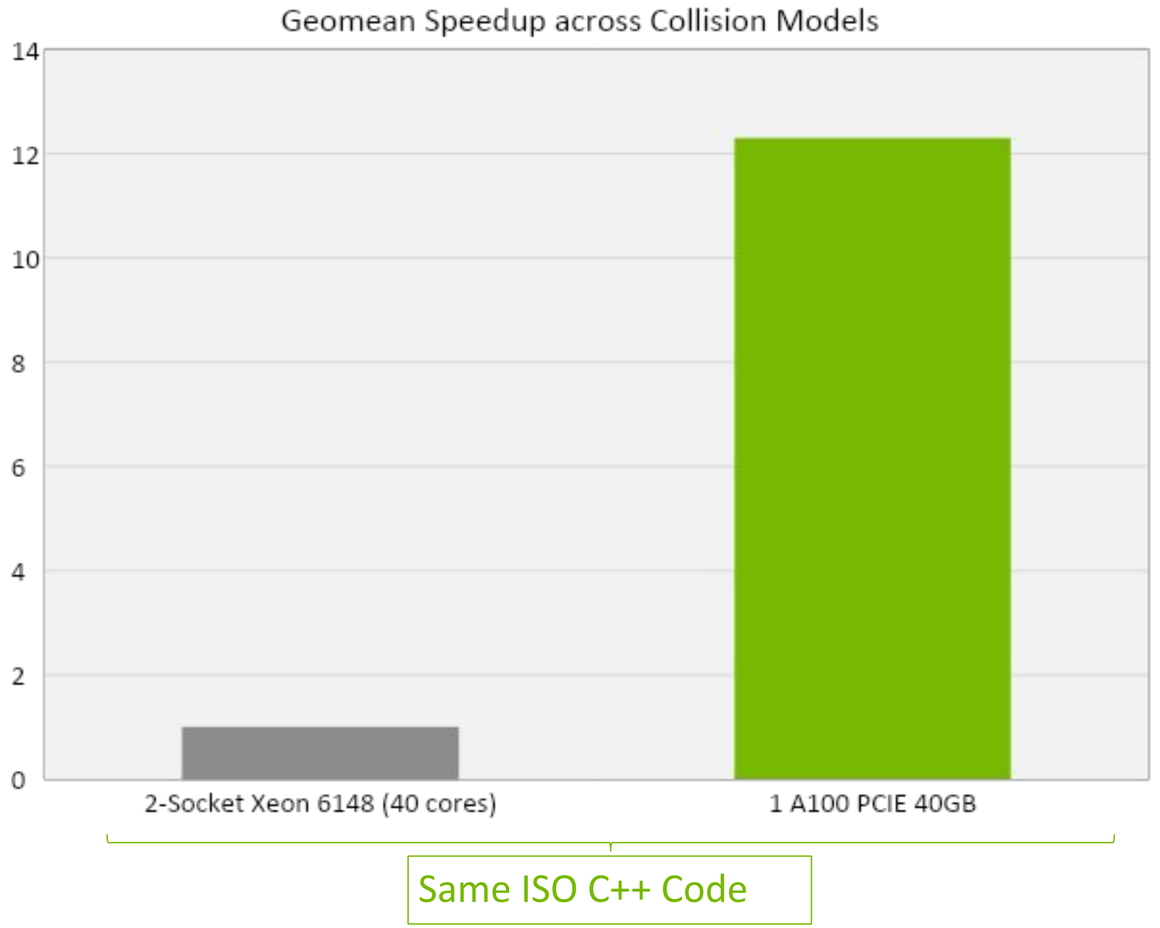
STLBM
Many-core Lattice Boltzmann with C++ Parallel Algorithms

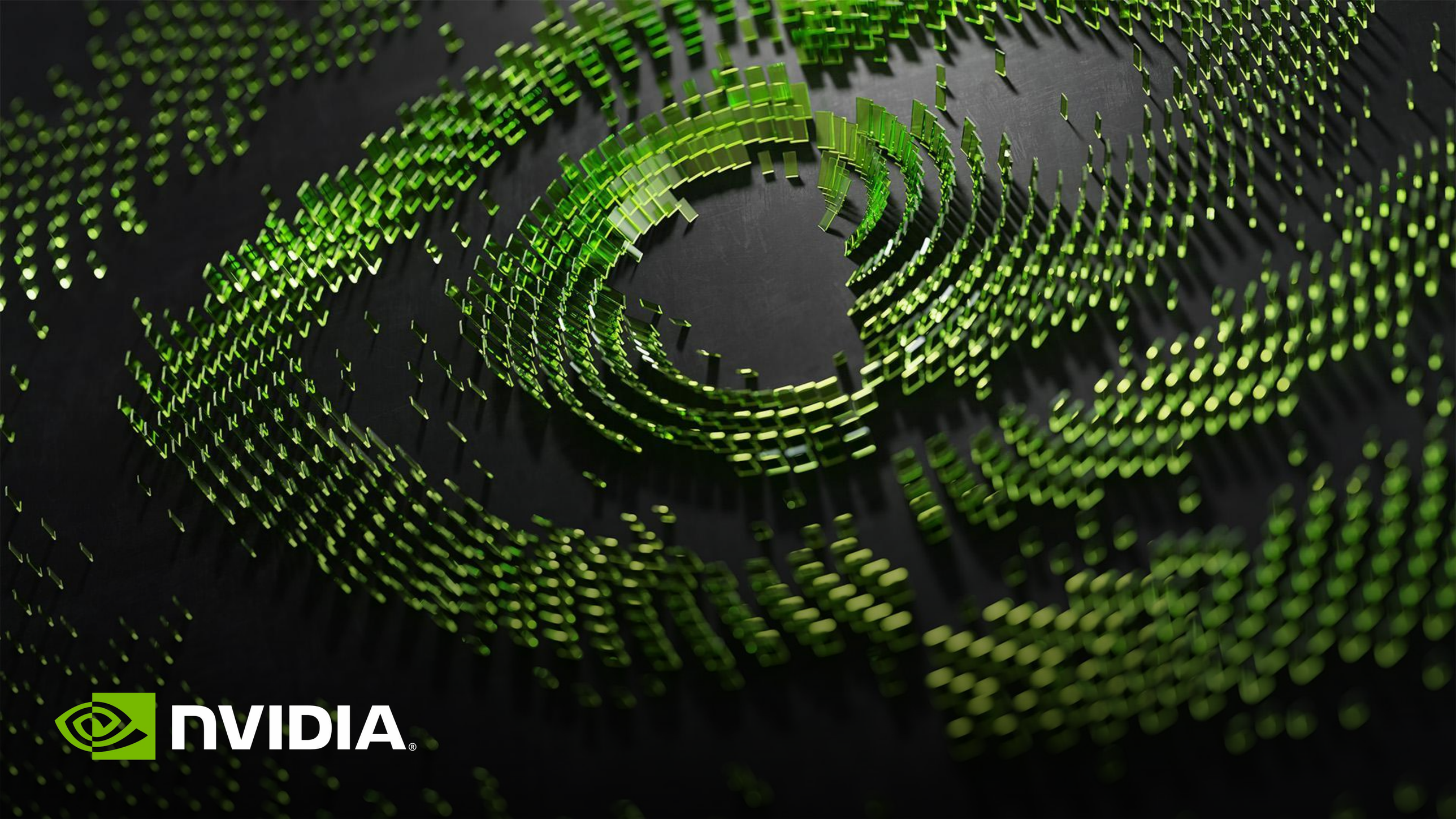
- Framework for parallel lattice-Boltzmann simulations on multiple platforms, including many-core CPUs and GPUs
- Implemented with C++17 standard (Parallel Algorithms) to achieve parallel efficiency
- No language extensions, external libraries, vendor-specific code annotations, or pre-compilation steps

"We have with delight discovered the NVIDIA "stdpar" implementation of C++ Parallel Algorithms. … We believe that the result produces state-of-the-art performance, is highly didactical, and introduces ***a paradigm shift in cross-platform CPU/GPU programming*** *in the community."*

-- Professor Jonas Latt, University of Geneva

https://gitlab.com/unigehpfs/stlbm
https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s32076/
GTC Fall Session A31329




NVIDIA