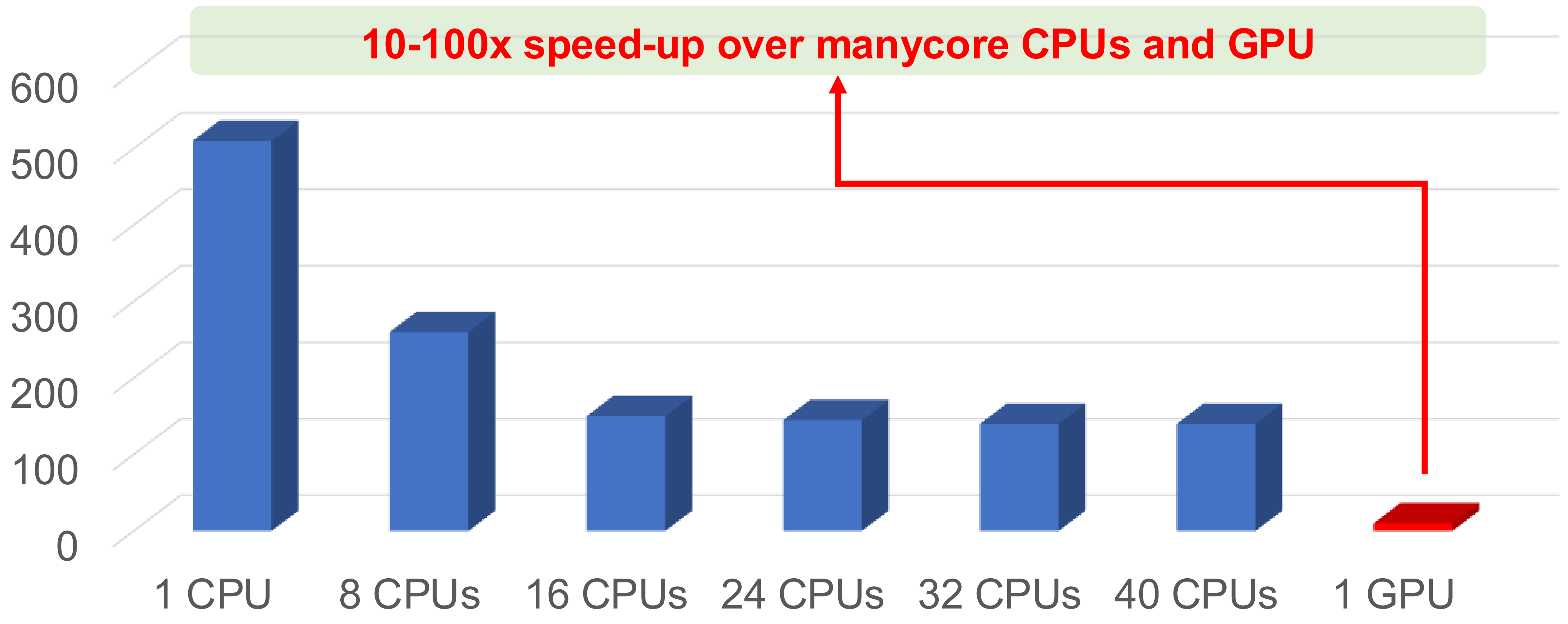
Taskflow: A General-purpose Task-parallel Programming System

Dr. Tsung-Wei (TW) Huang
Department of Electrical and Computer Engineering
University of Wisconsin at Madison, Madison, WI
https://taskflow.github.io/

Why Parallel Computing?

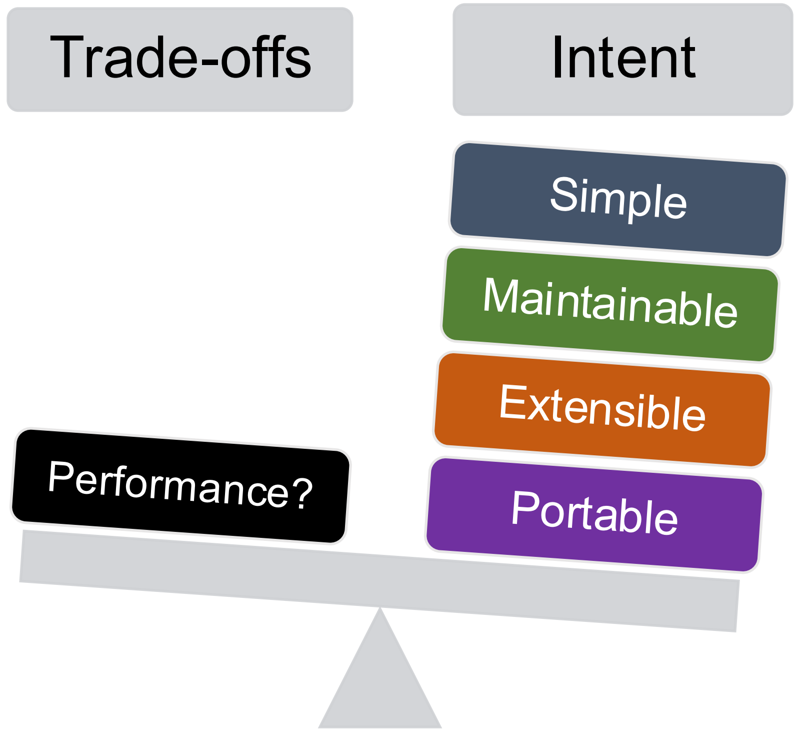
- **Advances performance to a new level previously out of reach**



Time (minutes) to train a machine learning model

Parallel Programming is Not Easy!

- **You need to deal with A LOT OF technical details**
  - Parallelism abstraction (software + hardware)
  - Concurrency control
  - Synchronization
  - Task and data race avoidance
  - Dependency constraints
  - Scheduling efficiencies (load balancing)
  - Programming productivity
  - Performance portability
  - …
- **And, don't forget about trade-offs**
  - Performance vs Developer's intent

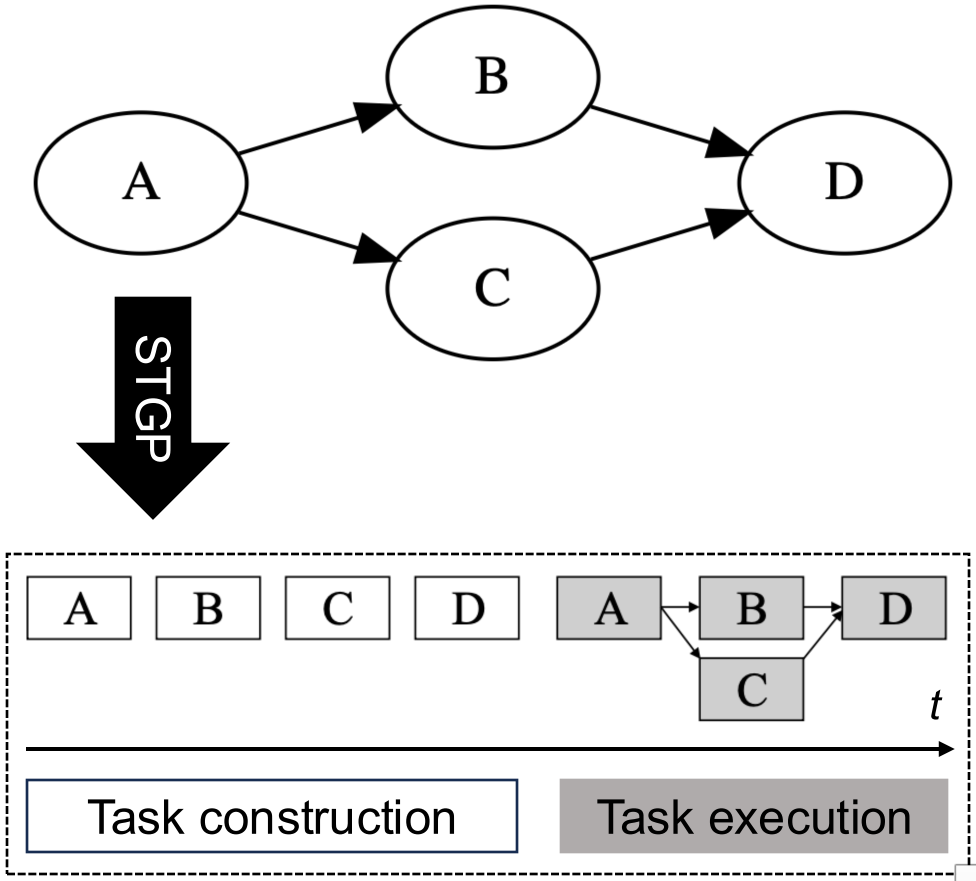


We want a solution that can sit on top to help programmers manage these details as much as possible because programmers care how fast (performance + productivity) they can get things done!

Static Task Graph Programming in Taskflow[1]

```
#include <taskflow/taskflow.hpp>  // Live: https://godbolt.org/z/j8hx3xnnx

int main(){
  tf::Taskflow taskflow;
  tf::Executor executor;
  auto [A, B, C, D] = taskflow.emplace(
    [] () { std::cout << "TaskA\n"; }
    [] () { std::cout << "TaskB\n"; },
    [] () { std::cout << "TaskC\n"; },
    [] () { std::cout << "TaskD\n"; }
  );
  A.precede(B, C);
  D.succeed(B, C);
  executor.run(taskflow).wait();
  return 0;
}
```



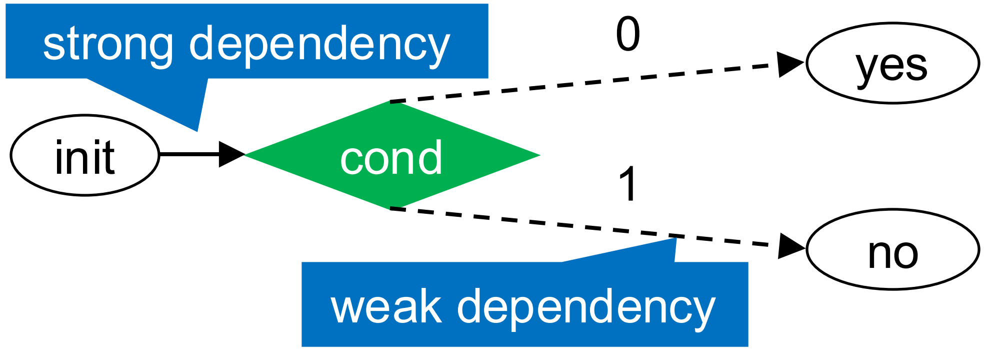
[1]: T.-W. Huang, et. al, "Taskflow: A Lightweight Parallel and Heterogeneous Task Graph Computing System," *IEEE TPDS*, 2022

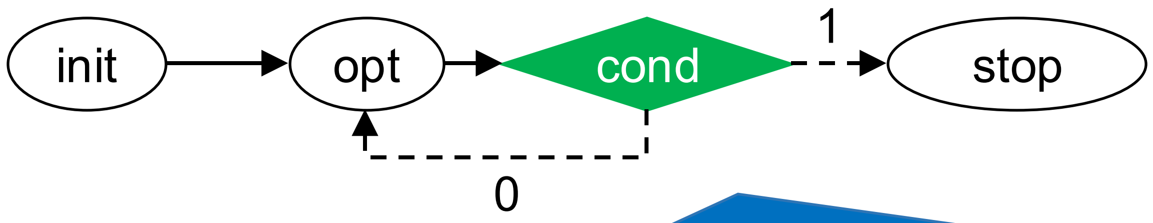
Control Taskflow Graph (CTFG) Programming Model

- **A key innovation that distinguishes Taskflow from existing libraries**

```
auto [init, cond, yes, no] =
taskflow.emplace(
  [] () { },
  [] () { return 0; },
  [] () { std::cout << "yes"; },
  [] () { std::cout << "no"; }
);
cond.succeed(init)
    .precede(yes, no);
```

```
auto [init, opt, cond, stop] =
taskflow.emplace(
  [&](){ initialize_data_structure(); },
  [&](){ some_optimizer(); },
  [&](){ return converged() ? 1 : 0; },
  [&](){ std::cout << ”done!\n"; }
);
opt.succeed(init).precede(cond);
converged.precede(opt, stop);
```





CTFG goes beyond the limitation of traditional DAG-based frameworks (no in-graph control flow).

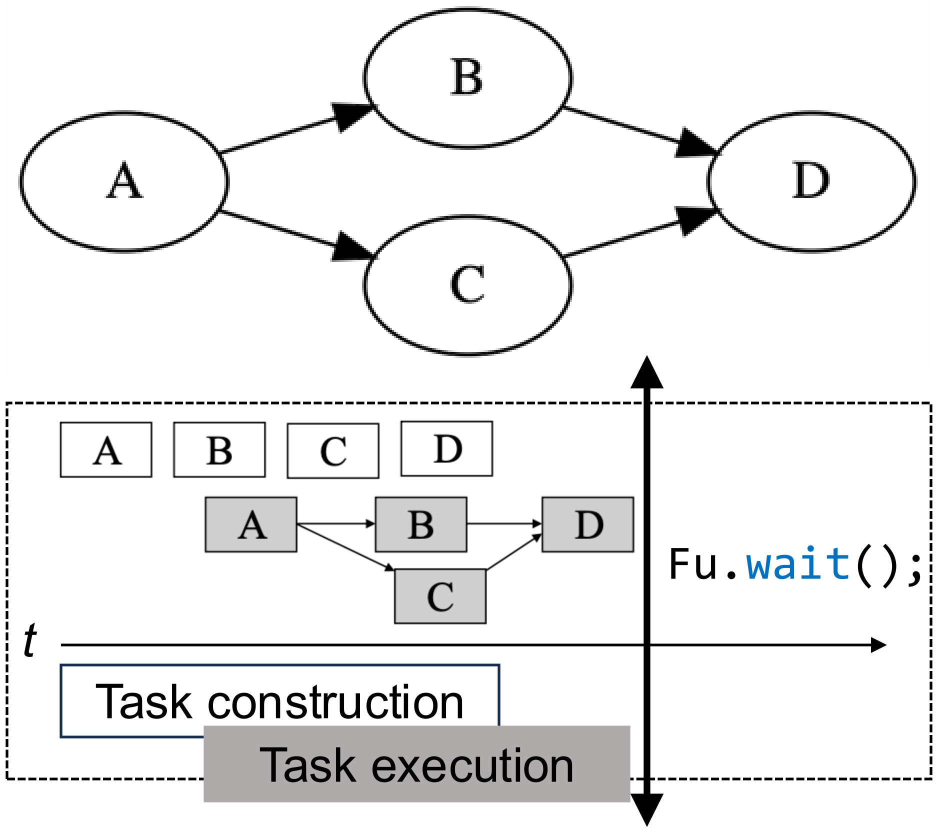
Dynamic Task Graph Programming in Taskflow

```
// Live: https://godbolt.org/z/j76ThGbWK

tf::Executor executor;

auto A = executor.silent_dependent_async([](){
    std::cout << "TaskA\n";
});
auto B = executor.silent_dependent_async([](){
    std::cout << "TaskB\n";
}, A);
auto C = executor.silent_dependent_async([](){
    std::cout << "TaskC\n";
}, A);
auto [D, Fu] = executor.dependent_async([](){
    std::cout << "TaskD\n";
}, B, C);

Fu.wait();
```



Specify arbitrary task dependencies on previously created tasks

Using Taskflow is EXTREMELY EASY

- **Taskflow is a header-only library built entirely with standard C++ libraries**
  - No wrangling with installation – just copy the headers and tell your compiler where to find them

```
# clone the Taskflow project
~$ git clone https://github.com/taskflow/taskflow.git
~$ cd taskflow
# compile your program and tell your compiler where to find Taskflow header files
~$ g++ -std=c++20 examples/simple.cpp –I ./ -O2 -pthread -o simple
~$ ./simple
TaskA
TaskC
TaskB
TaskD
```

- **Taskflow has been evolving over the years to a stable programming system**
  - Started in 2018 as a DARPA-sponsored research project to parallelize critical EDA applications

[1]: Taskflow: A General-purpose Task-parallel Programming system in Modern C++: https://taskflow.github.io/

Thank you for using Taskflow! https://taskflow.github.io/

OT ossia deal.II MYDATAMODELS XANADU LiveHD

ROS industrial REVEAL SNIG Inference Engine GROK 24h OPERON

R nVIDIA. Explosion METHANE Rapid Fuzz JET BRAINS RESEARCH atomicDEX

RAPIDS AMD TESSERACT COCOS MUNICH QUANTUM TOOLKIT

ModuleWorks Roche …