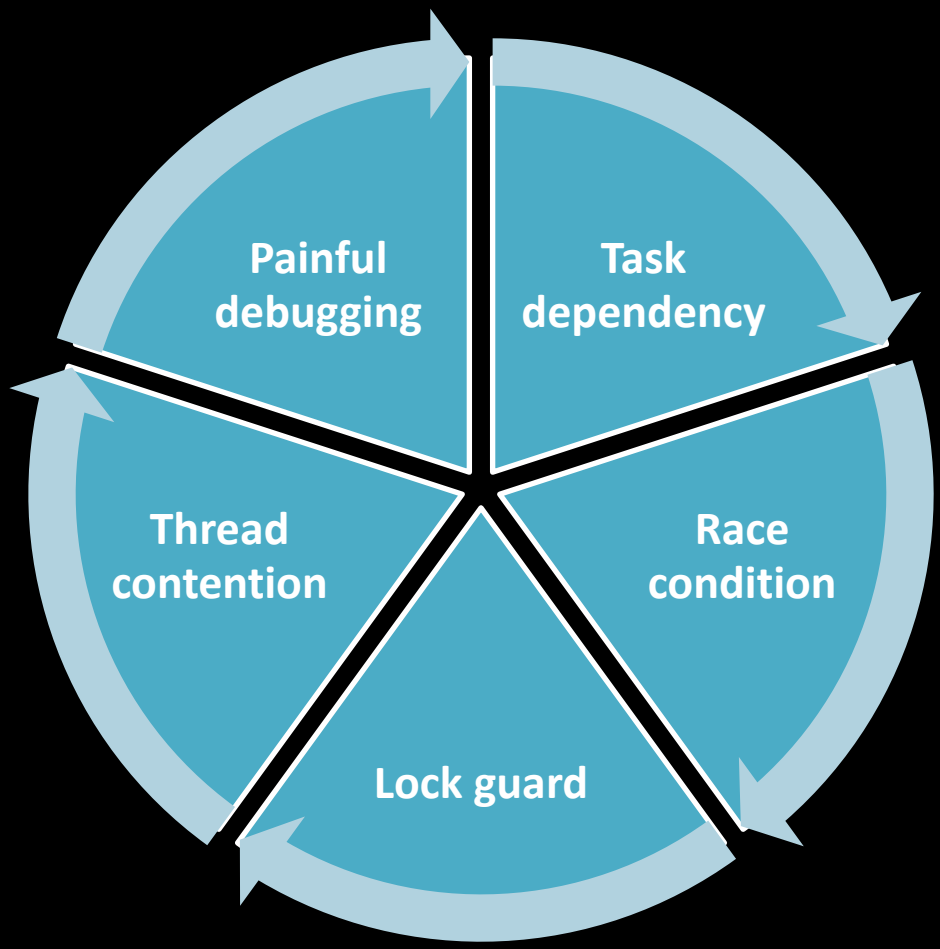
Fast Parallel Programming using Modern C++

Tsung-Wei Huang (twh760812@gmail.com)
The University of Illinois at Urbana-Champaign, IL, USA

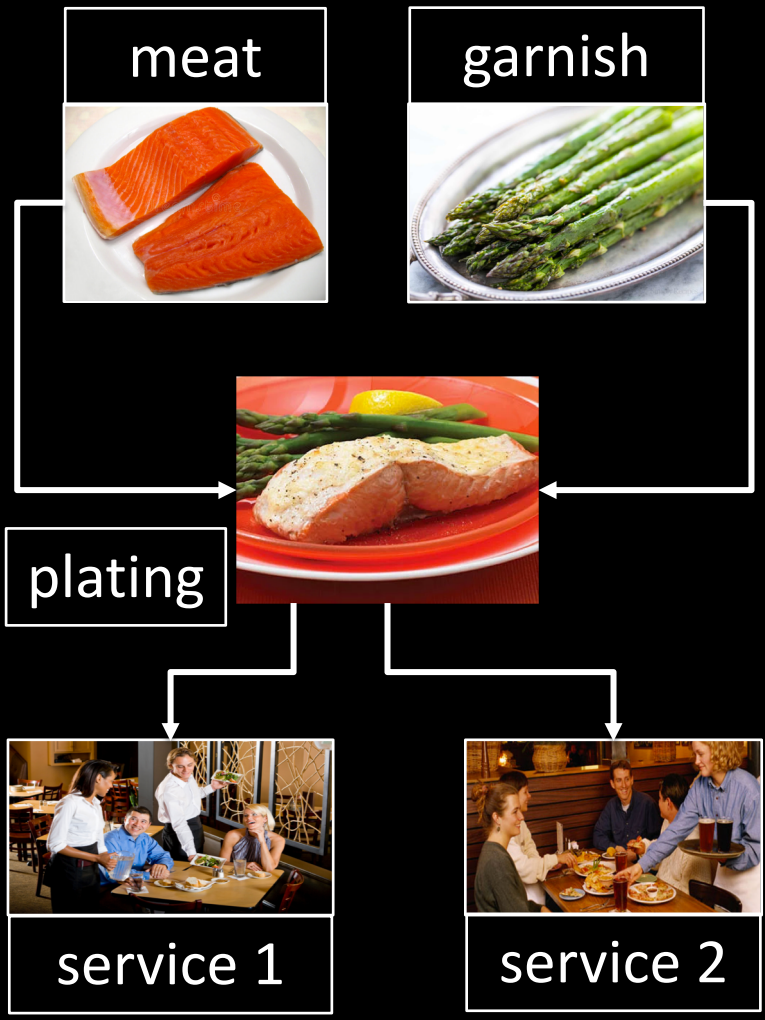
Cpp-Taskflow: https://github.com/cpp-taskflow/cpp-taskflow

Parallel Programming is NOT Trivial …



Hard-coded Parallelism

```
atomic<bool> garnish_ready {false};
atomic<bool> entree_ready  {false};
atomic<bool> plates_ready  {false};

thread cook1  ([&] {
  garnish = CookGarnish();
  garnish_ready = true;
});

thread cook2  ([&] {
  entree = CookEntree();
  entree_ready = true;
});

thread chief  ([&] {
  while(!(entree_ready && garnish_ready));
  plates = Plate(garnish, entree);
  plates_ready = true;
});

thread waiter1([&] {
  while(!plates_ready);
  Serve(plates.first);
});

thread waiter2([&] {
  while(!plates_ready);
  Serve(plates.second);
});
```

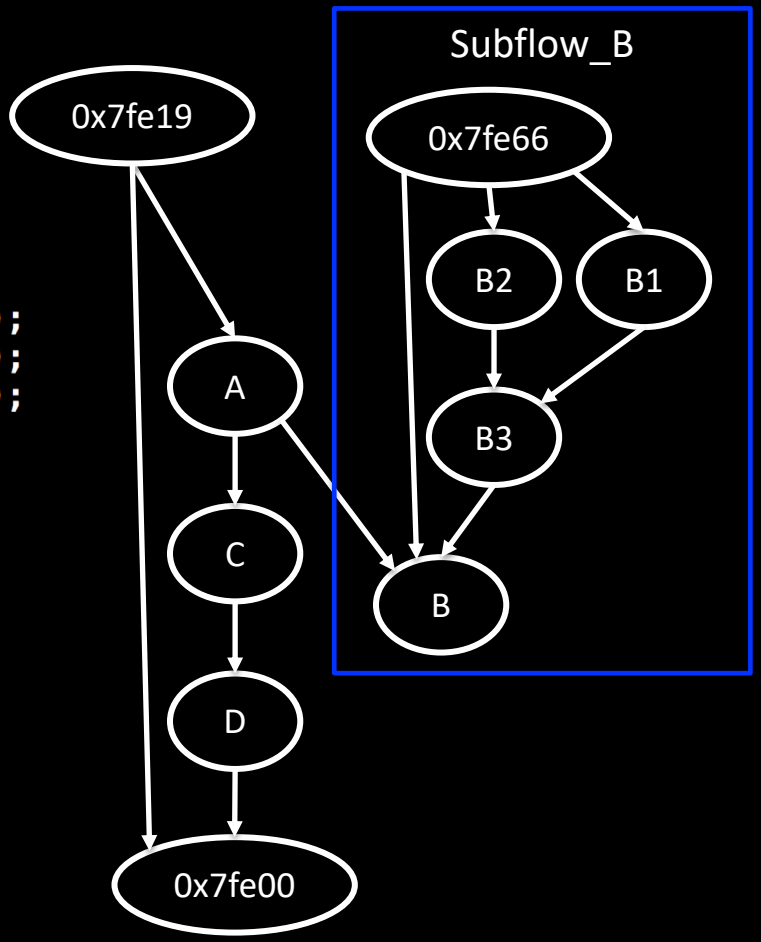


Cpp-Taskflow: Task-based Multi-threading Library

```
// create a taskflow object
Taskflow tf;

// create five tasks
auto [cook1, cook2, chief, waiter1, waiter2] = tf.silent_emplace(
  [&] () { garnish = CookGarnish();          },
  [&] () { entree  = CookEntree();           },
  [&] () { plates  = Plate(garnish, entree); },
  [&] () { Serve(plates.first);              },
  [&] () { Serve(plates.second);             }
);

// add dependencies
cook1.precede(chief);
cook2.precede(chief);
chief.precede(waiter1);
chief.precede(waiter2);

// execute
tf.wait_for_all();
```

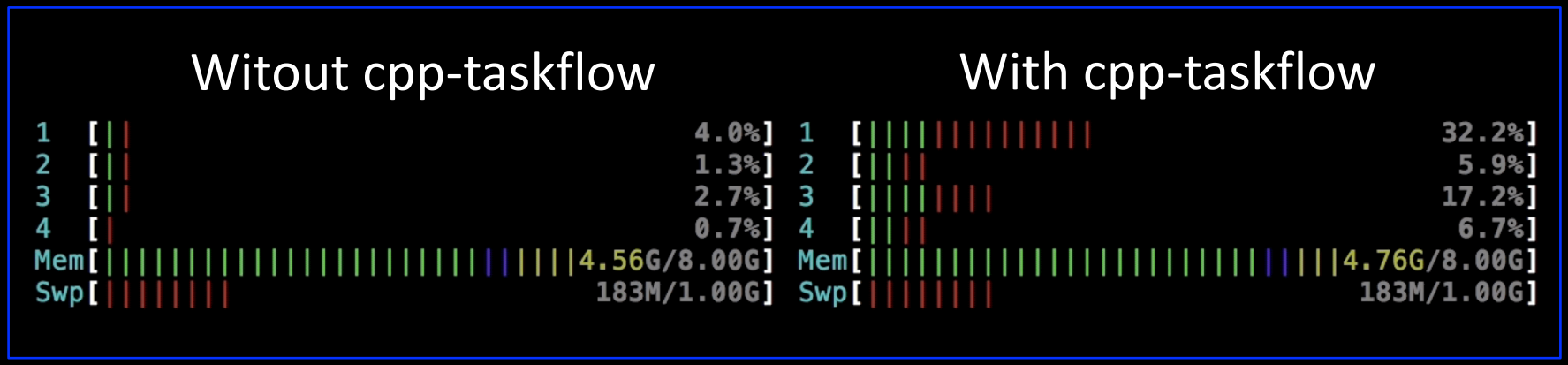
Dynamic Tasking

```
// create three regular tasks
auto A = tf.silent_emplace([](){}).name("A");
auto C = tf.silent_emplace([](){}).name("C");
auto D = tf.silent_emplace([](){}).name("D");

// create a subflow graph (dynamic tasking)
auto B = tf.silent_emplace([] (auto& subflow) {
  auto B1 = subflow.silent_emplace([](){}).name("B1");
  auto B2 = subflow.silent_emplace([](){}).name("B2");
  auto B3 = subflow.silent_emplace([](){}).name("B3");
  B1.precede(B3);
  B2.precede(B3);
}).name("B");

A.precede(B);  // B runs after A
A.precede(C);  // C runs after A
B.precede(D);  // D runs after B
C.precede(D);  // D runs after C

// execute the graph without cleanning up topologies
tf.dispatch().get();
cout << tf.dump_topologies();
```



Thank you!

Cpp-Taskflow: https://github.com/cpp-taskflow/cpp-taskflow



Acknowledgment: Chun-Xun Lin, Guannan Guo, and Martin Wong