Contech: Modeling and Analyzing Parallel Programs with Task Graphs

Brian Railing, Georgia Institute of Technology

What is Contech?

- Parallel Program Analysis Framework
- Compiler-based Parallel Program Instrumentation
- Open Source:
  http://bprail.github.io/contech/

Contech Requirements

- Task Graph analysis
  - Compiler with C++11 support
  - zlib (http://www.zlib.net/zlib.html)
- Contech Instrumentation
  - LLVM + Clang built with LTO support
    - Requires gold linker

Goals of Contech

- Provide a common representation for diverse parallel programs
- High performance instrumentation to generate this representation

Goals of Tutorial

- Learn the basics of Contech's task graph representation
- Explore the compiler-based instrumentation
  - Understand what it supports
  - What it does not support
  - And how to change these statements
- How to write program analyses with Contech

Parallel Program Diversity

- Language Diversity
- Runtime Diversity
- Pattern Diversity
- Platform Diversity

Diversity Survey

- What languages, runtimes, etc are you, the attendees, using?

Analyzing Parallel Programs

- (examples)

Analysis Support

- Analysis tools can target the program itself
  - Not implementation details
- Task graphs are a common representation
  - Agnostic of many program details
- C++11 API for accessing task graphs

Common Program Representation

- A common representation needs
  - What was executed
  - What was accessed
  - In what order did threads execute
- Without recording
  - Context switches
  - Consistency model
  - Cache Effects
  - …

Outline

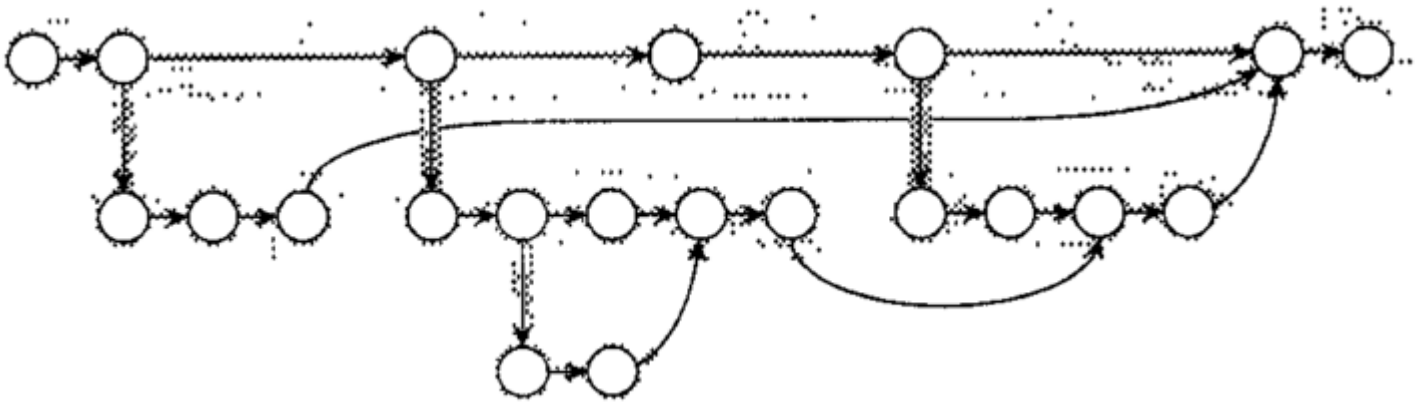
- Introduction
- **Contech's Task Graph Representation**
- Parallel Program Instrumentation
- (Break)
- Analysis and Usage of a Contech Task Graph
- Hands-on Exercises

Task Graph Representation

- A directed acyclic graph
  - Nodes are tasks, which describe “work”
  - Edges are dependencies between tasks

Prior Task Graph Work

- Originally, a representation for evaluating scheduling algorithms
  - Programs were abstract computation graphs



Blumofe , et al. STOC 1993

Prior Generation of Task Graphs

- Task graphs can also be used for runtime scheduling
- Language Choice
  - Cilk, HPF
- Program Structure
  - Regular Access / Execution Patterns
- Programmer Effort
  - Pragmas, Wrapper Routines

Contech Task Graph

- Generate the graph with no user intervention
  - Without constraint of language, library, or structure
- Task Graphs contain
  - Nodes partitioned based on type
  - Edges as scheduling dependencies
  - Nodes contain lists of actions and data
  - Other graph annotations such as start / end time

Contech Task Graph cont.

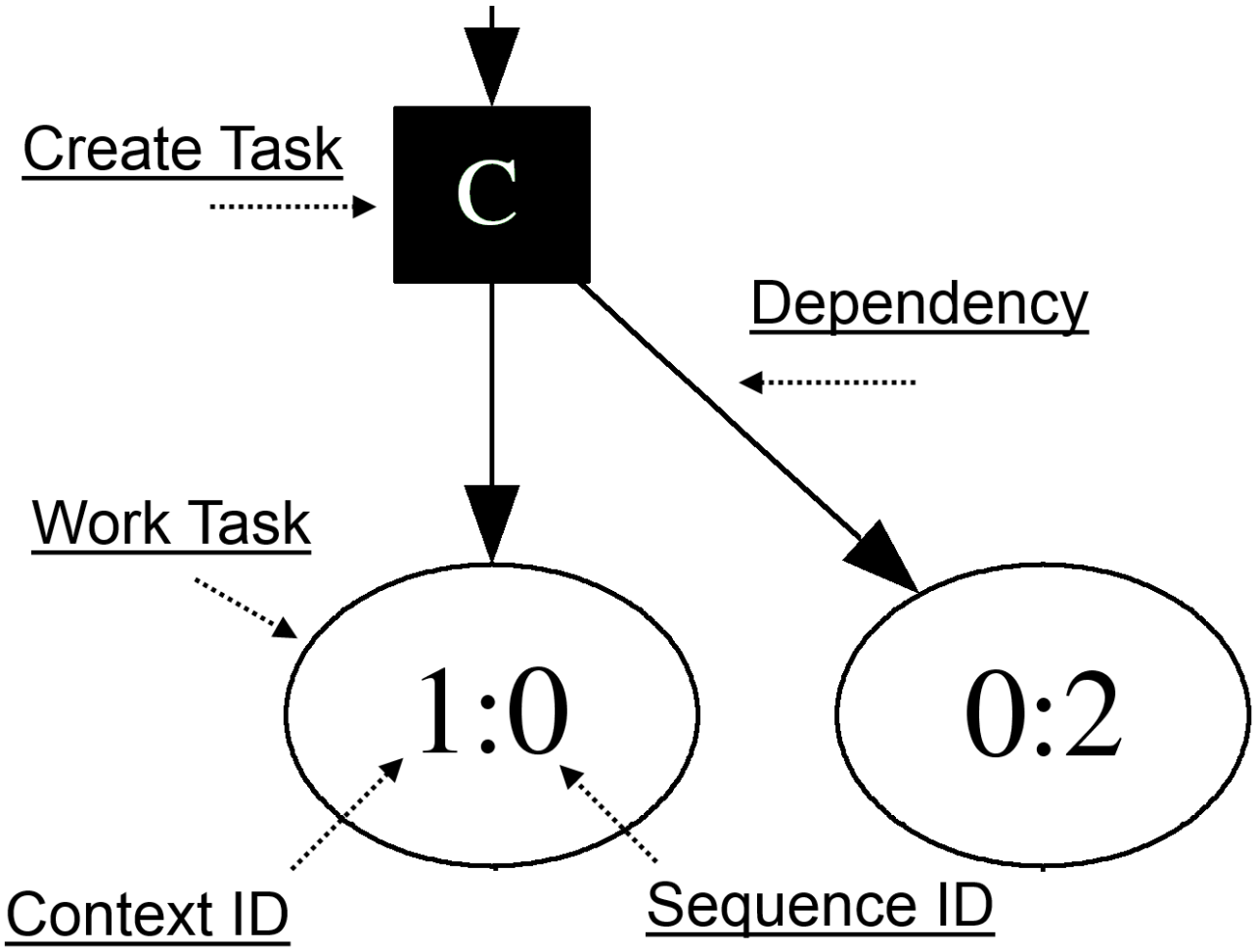
- Types of Nodes
  - 

  - Work (a.k.a. Basic Block)
- Edges are not partitioned, thus have no type

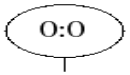
Node Identifiers

- Contech Task Graph Nodes have two identifiers
  - Context ID
    - Identifies an aggregation of concurrent work
    - Including: Thread, hardware context, task, loop iteration
  - Sequence ID
    - Where this task is ordered in its Context

Task Graph Legend

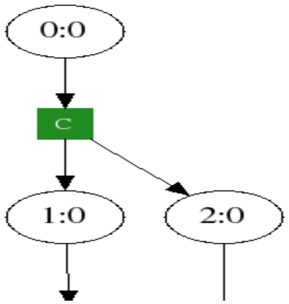


Task Graph Example




```
#pragma omp parallel
{
     #pragma omp single
     {
          ;
     }
     ;
}
```

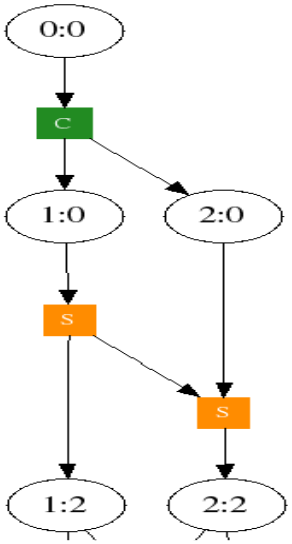
Task Graph Example




```
{
        #pragma omp single
        {
                ;
        }
        ;
}
```

Task Graph Example

```
#pragma omp parallel
{
        [redacted]
        {
                ;
        }
        ;
}
```

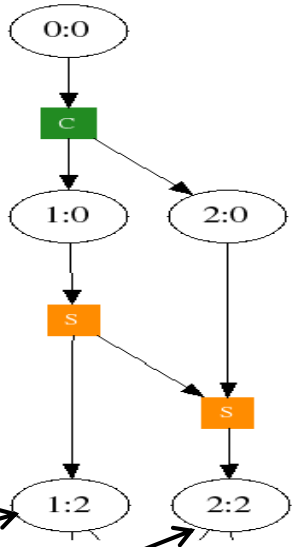


Task Graph Example

```
#pragma omp parallel
{
      #pragma omp single
      {
      }
      ;
}
```

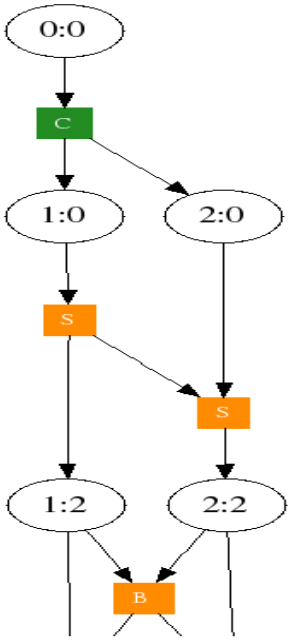
BBID X
BBID Y
BBID Z

BBID Z



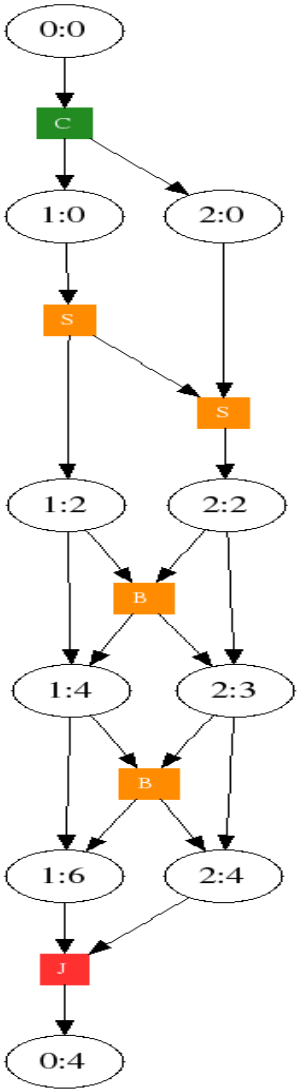
Task Graph Example

```
#pragma omp parallel
{
      #pragma omp single
      {
            ;
      }
      ;
}
```



Task Graph Example

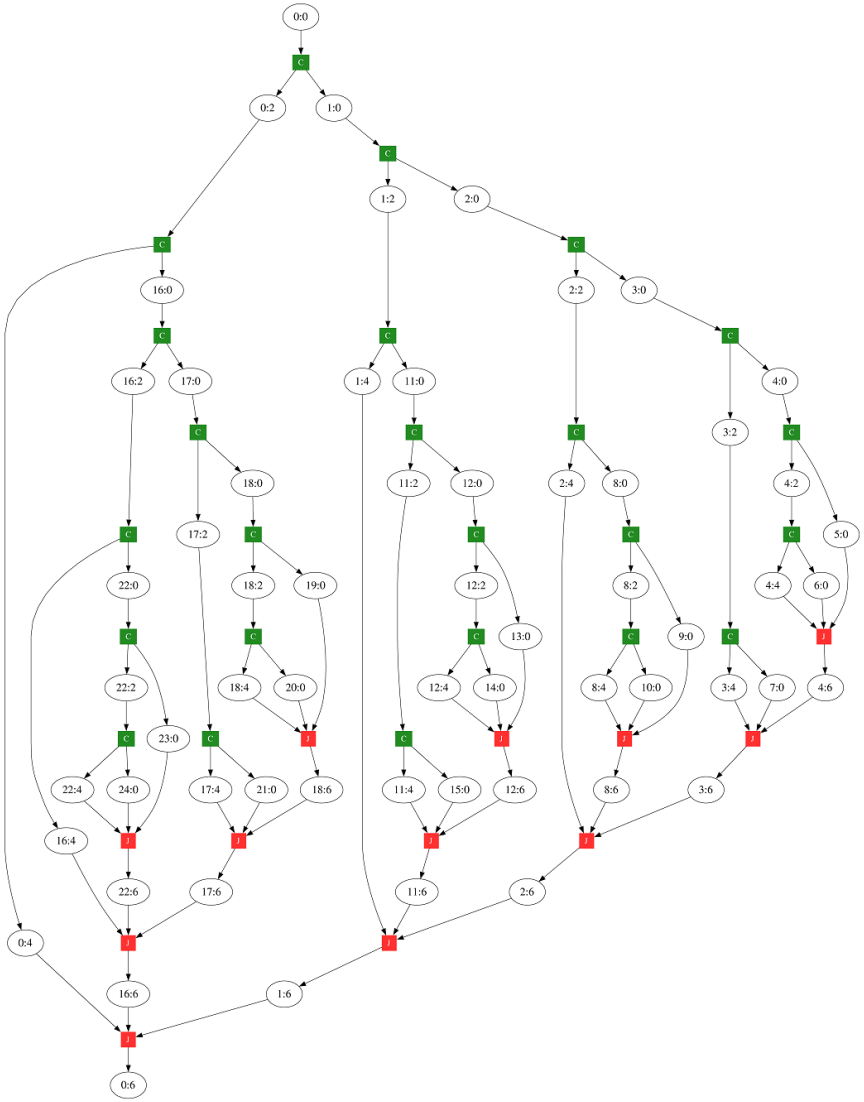
```
#pragma omp parallel
{
    #pragma omp single
    {
        ;
    }
    ;
```



Barrier Owner?

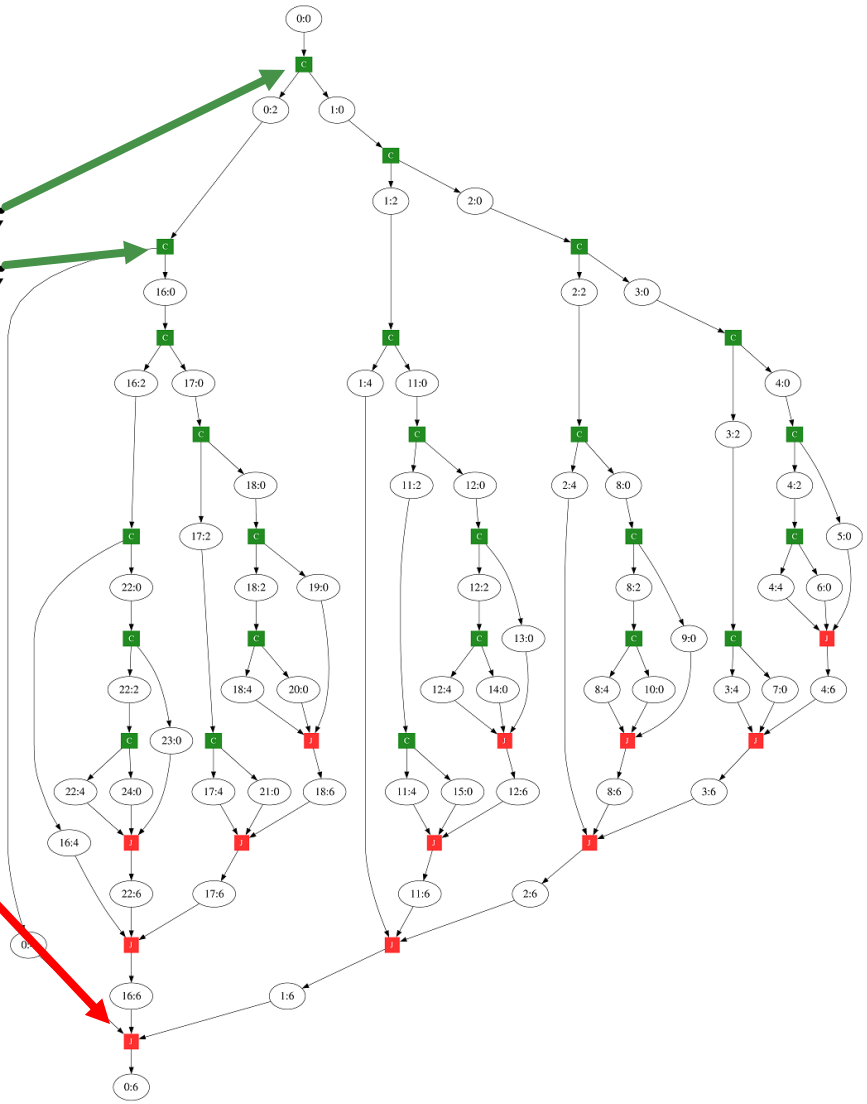
- All tasks are attributed to a Context
  - Which Context “owns” a barrier?
  - There is no right answer

Task Graph Example 2



Task Graph Example 2

```
int fib(int n) {
  if (n < 2)
    return n;
  int a = cilk_spawn fib(n-1);
  int b = cilk_spwan fib(n-2);
  cilk_sync;
  return a + b;
}
```

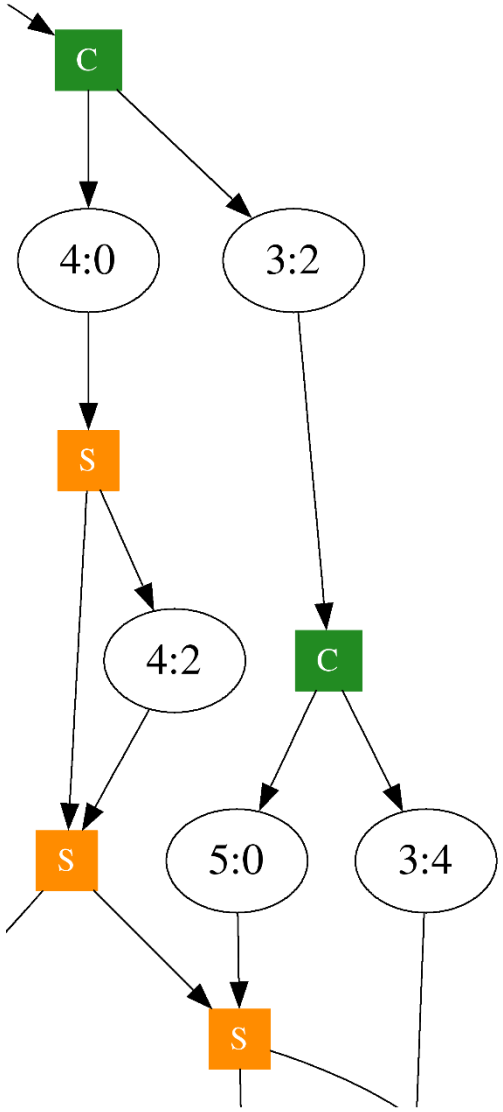


Empty Nodes

- Task graph construction alternates work and non-work in a Context
  - Certain cases result in the work task containing no work
  - This is an artifact of the implementation, not a fundamental invariant of the graph

OpenMP Task Example

- OpenMP Tasks
  - 4:0 is empty
    - Sync task is in/out dependency
  - 4:2 is body of task
  - 5:0 is empty
    - Dependent on result from Context 4



Summary

- Contech's Task Graph representation
  - Unifies diverse parallel programs into common format
  - Provides independence from the architecture and implementation details