Concurrency in LLVM

George Stelle
Amalee Wilson
Alexis Perry-Holby
Patrick McCormick

Los Alamos National Laboratory

February 28, 2019

Outline

- Motivation
- Parallel IR
- Frontends and Backends
- Labeled Concurrency
- Compiler Pipelining
- Compilation Benefits

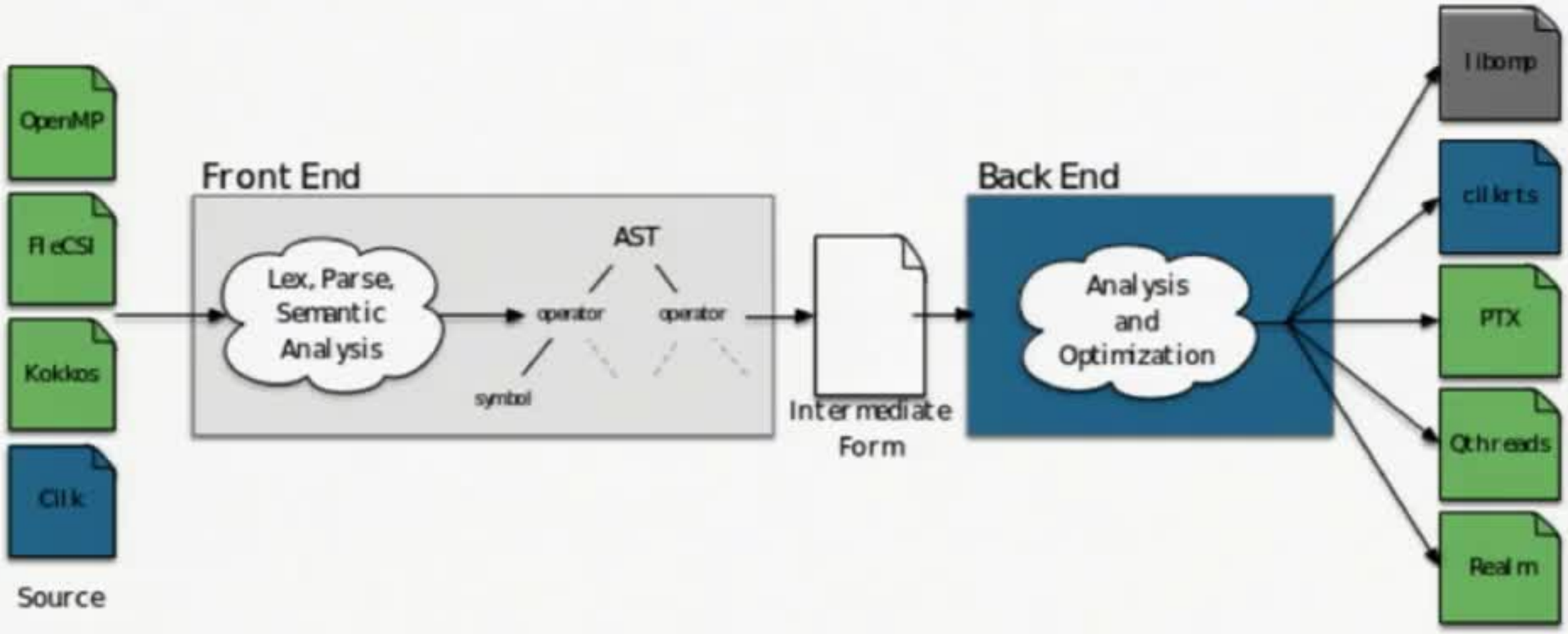
Motivation

```
...
  %19 = call i32 @__kmpc_omp_task(%ident_t* nonnull @0, i32
  %20 = call i8* @__kmpc_omp_task_alloc(%ident_t* nonnull @0
  %21 = bitcast i8* %20 to i8**
  %22 = load i8*, i8** %21, align 8, !tbaa !8
...
  %28 = load i32, i32* %2, align 4, !tbaa !4
  store i32 %28, i32* %27, align 4, !tbaa !13
  %29 = call i32 @__kmpc_omp_task(%ident_t* nonnull @0, i32
  %30 = call i32 @__kmpc_omp_taskwait(%ident_t* nonnull @0,
...
```

Parallel IR (Tapir)

```
w
|\---detach s x y
| x
|/---reattach s y
y
|----sync s z
z
```

Frontends and Backends



Labeled Tasks

Nested Tasks

OpenMP

```
#omp task
#omp taskwait
```

Cilk

```
cilk_spawn
cilk_sync
```

Labeled Tasks

```
spawn x
spawn y
sync x
sync y
```

Compiler Pipelining

```
sync a;
gather(buf_a);
spawn a {
  compute(buf_a);
  scatter(buf_a);
}
sync b;
gather(buf_b);
spawn b {
  compute(buf_b);
  scatter(buf_b);
}
```