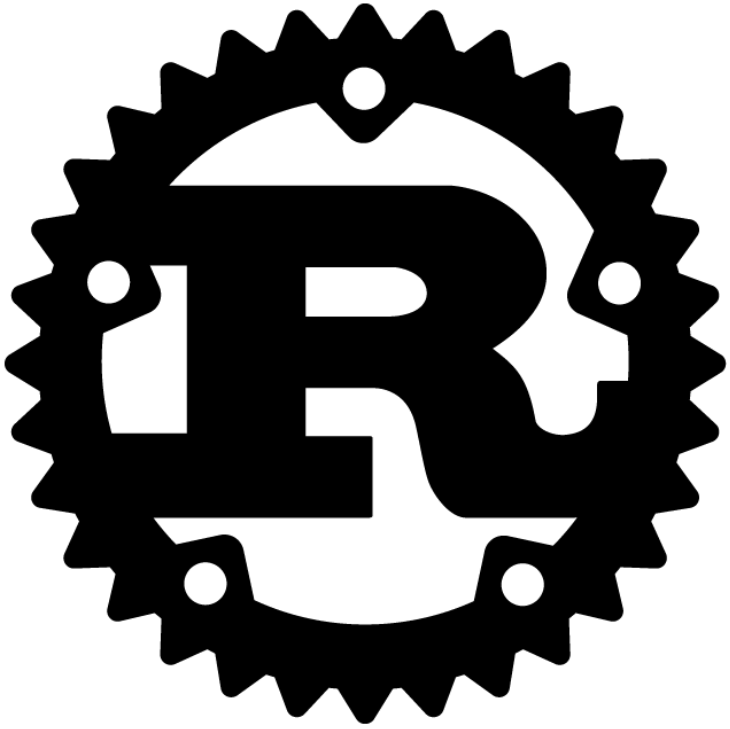


Compiling Rust for GPUs

Eric Holk and Milinda Pathirage

B649: Parallel Architectures and Systems

The Goal

```
#[kernel]
fn add_vectors(x: &[float], y: &[float], z: &[mut float]) {
    let i = GPU::thread_id();
    z[i] = x[i] + y[i];
}

fn main() {
    let A = random_vector(1000);
    let B = random_vector(1000);
    let C = zero_mut_vector(1000);

    kernel!(add_vectors, A, B, C);

    io::println(fmt!("%?", C));
}
```

Overview

- Introduction to Rust
- Adding GPU Support to Rust
- Working with the NVPTX LLVM Backend
- Current Status

Rust

a systems language

pursuing the trifecta

safe, concurrent, fast

-lindsey kuper

Pattern Matching

```
enum Direction {
    up, down, left, right
}

fn to_str(d: Direction) -> ~str {
    match d {
    up  => ~"up",
    down  => ~"down",
    left  => ~"left",
    right => ~"right"
    }
}
```

Polymorphism

```
enum Tree<T> {
    Leaf(T), Node(~Tree<T>, ~Tree<T>)
}

fn traverse<T>(t: ~Tree<T>, f: fn(T)) {
    match t {
    ~Leaf(x) => f(x),

    ~Node(left_child, right_child) => {
    traverse(left_child, f);
    traverse(right_child, f)
    }
    }
}
```

Bounded Polymorphism

```
trait ToStr { fn to_str() -> ~str; }

fn print_leaves<T: ToStr>(t: ~Tree<T>) {
    match t {
        ~Leaf(x) => io::println(x.to_str()),

        ~Node(left_child, right_child) => {
        print_leaves(left_child);
        io::println(" ");
        print_leaves(right_child)
        }
    }
}
```

Macros

```
macro_rules! trace (
    { $e:expr } => {
    if debug_enabled() {
        log(error, $e)
    }
    }
);

trace!(something_expensive());
```

Concurrency

- Lightweight, shared nothing tasks
- Message passing
- Send transfers ownership

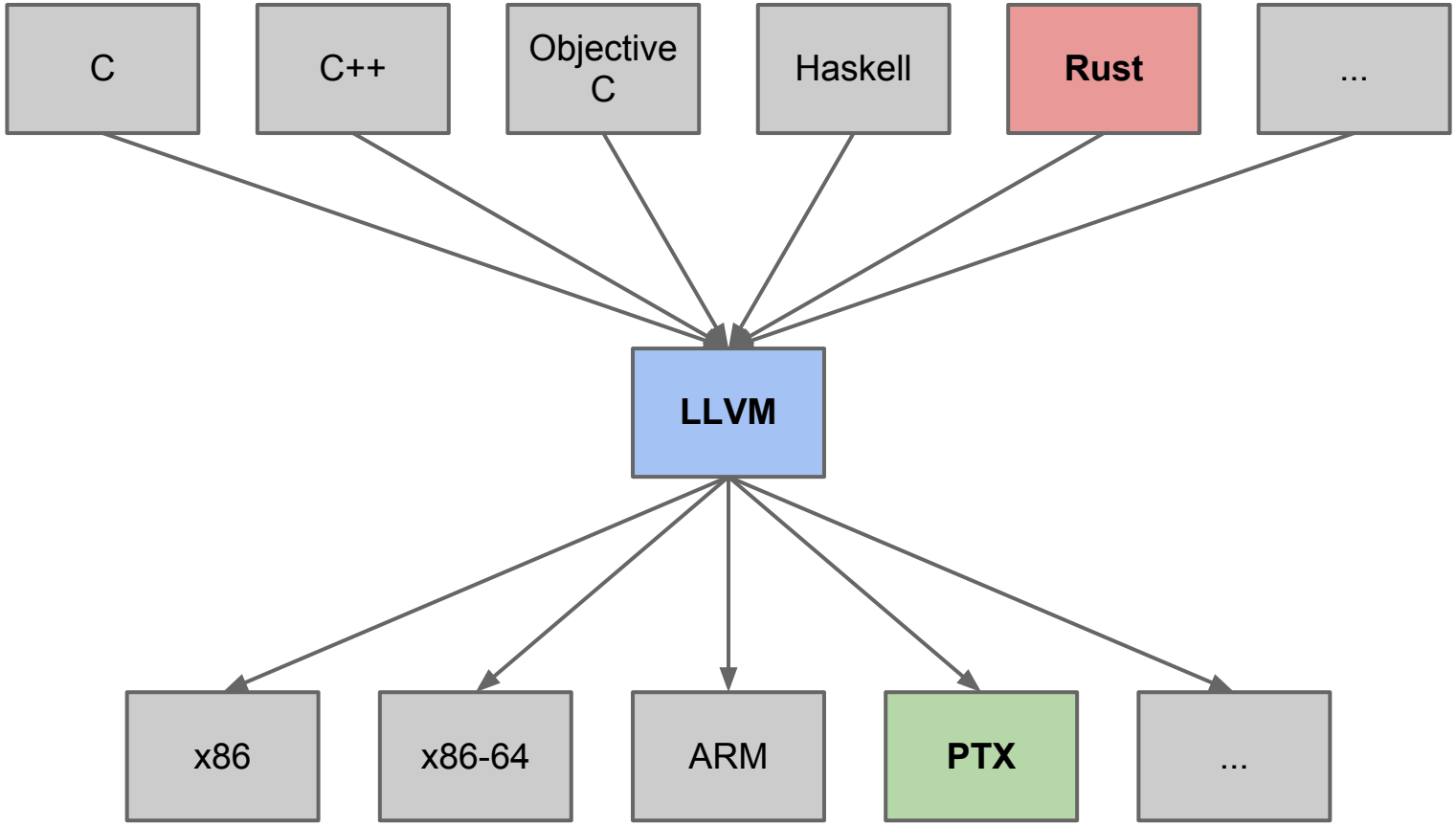
Adding GPU Support to Rust

- Annotate kernel functions with `#[kernel]`
- Compile with `-Zptx` to generate kernel
- Use OpenCL to load and execute the kernel
- Rust library code simplifies handling of data

The NVPTX Backend

- Produced by NVIDIA
- Part of LLVM trunk
- Used by CUDA since version 4.1
- Translates LLVM to PTX, NVIDIA's virtual assembly language for GPUs

LLVM Architecture



Current Status

- Basic examples work (with some care)
- Fixed several bugs in the NVPTX backend
  - Contributed fixes back to LLVM trunk
- Greatly improved Rust OpenCL bindings

Example (Rust)

```
#[kernel]
fn add_float(x: &float, y: &float, z: &mut float) {
    *z = *x + *y
}
```

Example (LLVM)

```
define ptx_kernel void @_ZN9add_float17_d08d41c0c85935643_00E(i1
addrspace(1)* nocapture, { i64, %tydesc*, i8*, i8*, i8 } addrspace(1)*
nocapture, double addrspace(1)* nocapture, double addrspace(1)*
nocapture, double addrspace(1)* nocapture) nounwind uwtable {
static_allocas:
  %5 = load double addrspace(1)* %2, align 8
  %6 = load double addrspace(1)* %3, align 8
  %7 = fadd double %5, %6
  store double %7, double addrspace(1)* %4, align 8
  ret void
}
```

```
.entry _ZN9add_float17_d08d41c0c85935643_00E(
    .param .u32 .ptr .global .align 1 _ZN9add_float17_d08d41c0c85935643_00E_param_0,
    .param .u32 .ptr .global .align 8 _ZN9add_float17_d08d41c0c85935643_00E_param_1,
    .param .u32 .ptr .global .align 8 _ZN9add_float17_d08d41c0c85935643_00E_param_2,
    .param .u32 .ptr .global .align 8 _ZN9add_float17_d08d41c0c85935643_00E_param_3,
    .param .u32 .ptr .global .align 8 _ZN9add_float17_d08d41c0c85935643_00E_param_4
)
{
    .reg .pred %p<396>;
    .reg .s16 %rc<396>;
    .reg .s16 %rs<396>;
    .reg .s32 %r<396>;
    .reg .s64 %rl<396>;
    .reg .f32 %f<396>;
    .reg .f64 %fl<396>;

    ld.param.u32     %r0, [_ZN9add_float17_d08d41c0c85935643_00E_param_3];
    ld.global.f64     %fl0, [%r0];
    ld.param.u32     %r0, [_ZN9add_float17_d08d41c0c85935643_00E_param_2];
    ld.global.f64     %fl1, [%r0];
    add.f64     %fl0, %fl1, %fl0;
    ld.param.u32     %r0, [_ZN9add_float17_d08d41c0c85935643_00E_param_4];
    st.global.f64     [%r0], %fl0;
    ret;
}
```

Questions?