**LLVM** and **clang**

- The **LLVM** Project is a collection of modular and reusable compiler and toolchain technologies (llvm.org).

- **clang** is a hackable C compiler built upon LLVM
Why make an **LLVM** backend for Chapel?

- Today, Chapel generates C
- After that, who knows what will happen?
  - Resulting speed?
  - Optimizations?
- Control it with LLVM
How does it work?

- **clang** compiles Chapel runtime (written in C) into an LLVM module
- Chapel code generator adds LLVM to module
- can call C functions
- can use C data types
C Integration
C Integration Today

```c
int foo(int);

int foo(int x) {
    return x + 1;
}
```

```
$ chpl foo.chpl foo.h foo.c
```
C through LLVM (1)

```
int foo(int);   int foo(int x) { return x + 1; }

extern "C" {
    #include "foo.h"

    writeln(foo(4));
}

```

```
$ chpl foo.chpl foo.c
```
extern "C" {
    int foo(int x) {
        return x + 1;
    }
}
writeln(foo(4));

$ chpl foo.chpl
Optimization
Example Optimization

```javascript
// x is remote
var sum = 0;
for i in 1..100 {
    sum += get(x);
}
```
// x is remote
var sum = 0;
for i in 1..100 {
  sum += get(x);
}

// existing LLVM opt
var sum = 0;
$r1 = get(x);
for i in 1..100 {
  sum += $r1;
}

TO GLOBAL MEMORY

var sum = 0;
for i in 1..100 {
  sum += load(x);
}

EXISTING LLVM OPTIMIZATION

// existing LLVM opt
var sum = 0;
$r1 = load(x);
for i in 1..100 {
  sum += $r1;
}

TO DISTRIBUTED MEMORY
Acknowledgements

- Matt Lentz and Joe Yan (UMD) - for helping to create the LLVM backend
- Chapel developers - for putting up with my huge patch
Thanks!

* LLVM is in Chapel v1.6!
* Finishing described optimization and C integration
* Do a better job with LLVM optimization
* type-based alias analysis
* forall -> SIMD/ILP