



LLVM



**Michael Ferguson**

**Laboratory for Telecommunications Sciences**



# *LLVM* and *clang*

- \* The **LLVM** Project is a collection of modular and reusable compiler and toolchain technologies ([llvm.org](http://llvm.org)).
- \* **clang** is a *hackable* C compiler built upon LLVM



PHOTO CREDIT: WALTERS ART MUSEUM



# 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



CAT CREDIT: TESLA



# How does it work?

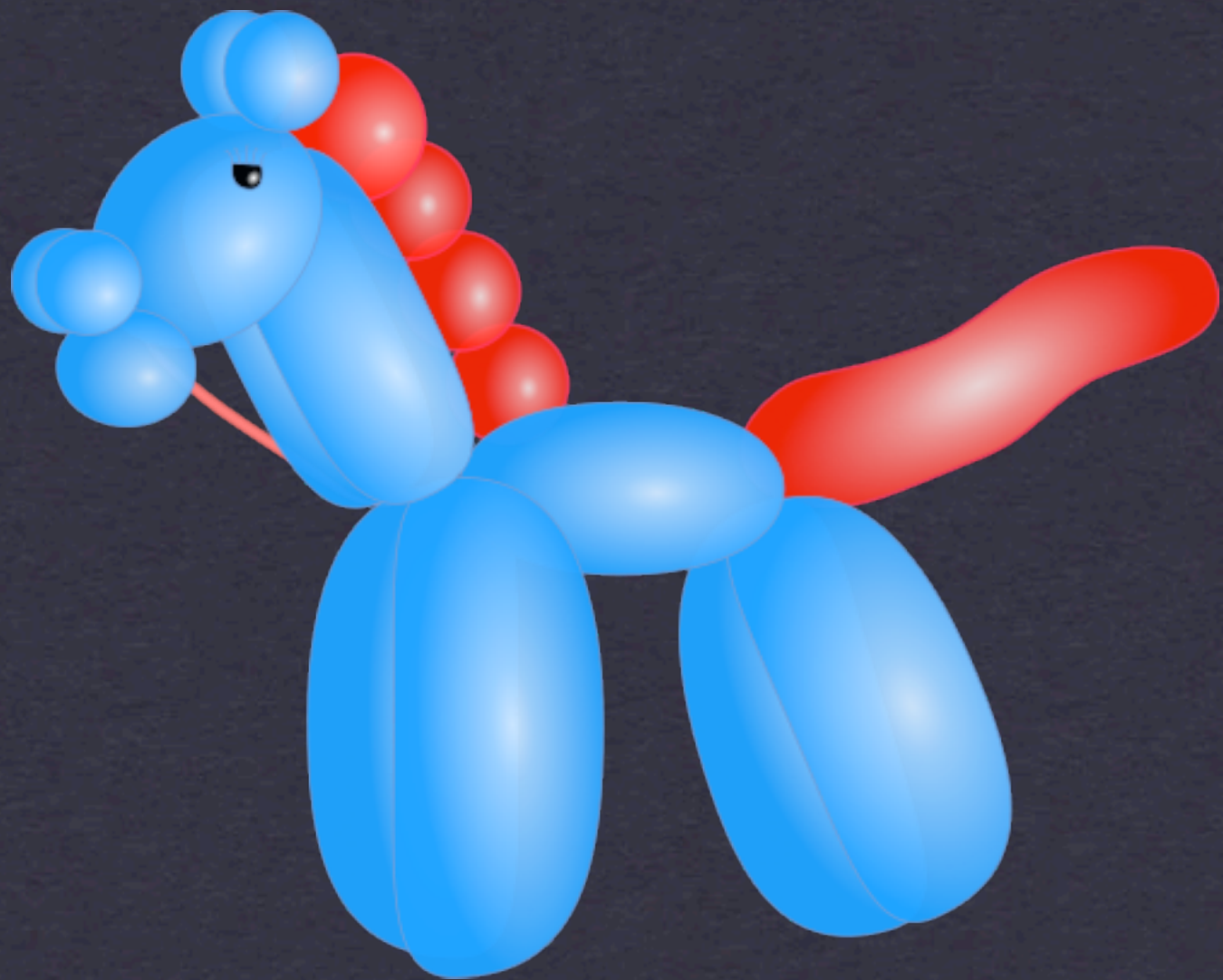
- ✱ ***clang*** compiles Chapel runtime (written in C) into an LLVM module
- ✱ Chapel code generator adds LLVM to module
- ✱ can call C functions
- ✱ can use C data types



PHOTO CREDIT: WALTERS ART MUSEUM



# C Integration





# C Integration Today

<code>int foo(int);</code>	<code>int foo(int x) { return x+1; }</code>
<b>FOO.H</b>	<b>FOO.C</b>

`extern proc foo(x:int):int;`

`writeln(foo(4));`  
**FOO.CHPL**

`$ chpl foo.chpl foo.h foo.c`



# C through LLVM (1)

<code>int foo(int);</code>	<code>int foo(int x) { return x+1; }</code>
<b>FOO.H</b>	<b>FOO.C</b>

```
extern "C" {  
    #include "foo.h"  
}  
writeln(foo(4));  
FOO.CHPL
```

```
$ chpl foo.chpl foo.c
```



# C through LLVM (2)

```
extern "C" {  
    int foo(int x) {  
        return x+1;  
    }  
}  
writeln(foo(4));  
FOO.CHPL
```

```
$ chpl foo.chpl
```



# Optimization





# Example Optimization

// x is remote

var sum = 0;

for i in 1..100 {

    sum += get(x);

}



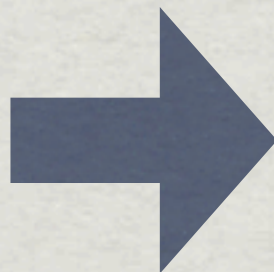
```
// x is remote  
var sum = 0;  
for i in 1..100 {  
    sum += get(x);  
}
```

TO GLOBAL  
MEMORY



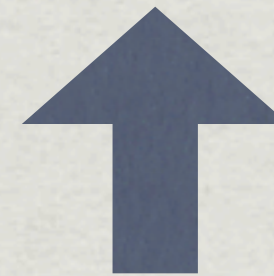
```
var sum = 0;  
for i in 1..100 {  
    sum += load(x);  
}
```

EXISTING LLVM  
OPTIMIZATION



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

TO DISTRIBUTED  
MEMORY



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



# 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

