Generating LLVM IR using Template Meta Programming

Timo von Holtz

September 18, 2014

Accelerate

- Embedded array language
- Compiled online
- CUDA backend

LLVM Backend

- CPU/GPU
- no external process
- incomplete

Overview LLVM

- Compiler library
- Portable
- Haskell bindings (llvm-general)
- Auto-Vectorization
- Static Single Assignment (SSA)

Code Example LLVM

```
int abs(int x) {
    int res;
    if (x>0) {
        res = x;
    } else {
        res = -x;
    }
    return res;
}
```

```
define i32 @abs(i32 %x) {
entry:
  %1 = icmp sgt i32 %x, 0
  br i1 %1, label %then, label %else

then:                                             ; preds = %entry
  br label %5

else:                                             ; preds = %entry
  %4 = sub nsw i32 0, %x
  br label %5

end:                                              ; preds = %else, %then
  %res.0 = phi i32 [ %x, %then ],
                   [ %4, %else ]
  ret i32 %res.0
}
```

llvm-general

- uses ADT to represent syntax tree
- translates from and to C++ Objects
- exposes Optimizer/JIT compiler

Quasiquoting

```
abs :: Global
abs = [llg|
  define i32 @abs(i32 %x) {
  entry:
    %1 = icmp sgt i32 %x, 0
    br i1 %1, label %end, label %else

  else:
    %4 = sub nsw i32 0, %x
    br label %5

  end:
    %res.0 = phi i32 [ %x, %entry ],
                     [ %4, %else ]
    ret i32 %res.0
  }
  |]
```

Control Structures

```
i32 foo(i32 %m, i32 %n) {
  entry:
    br label %for

  for:
    for i32 %i in %m to %n with i32 [ 0, %entry ] as %j,
                                                label %end {
        %k = add i32 %i, %j
        ret i32 %k
    }

  end:
    ret i32 %j
}
```

Control Structures

```
i32 foo(i32 %m, i32 %n) {
  entry:
    br label %for

  for:
    for i32 %i in %m to %n with i32 [ 0, %entry ] as %j {
        %k = add i32 %i, %j
        ret i32 %k
    }

  end:
    ret i32 %j
}
```

Control Structures

```
define i64 @foo(i64 %start, i64 %end) {
  entry:
    %x = i64 0

  for:
    for i64 %i in %start to %end {
        %x = add i64 %i, %x
    }

  exit:
    ret i64 %x
}
```

SSA

Braun, Matthias et al.: Simple and Efficient Construction of Static Single Assignment Form. In: Jhala, Ranjit ; Bosschere, Koen (eds.): *Compiler Construction*. Vol. 7791. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 102–122

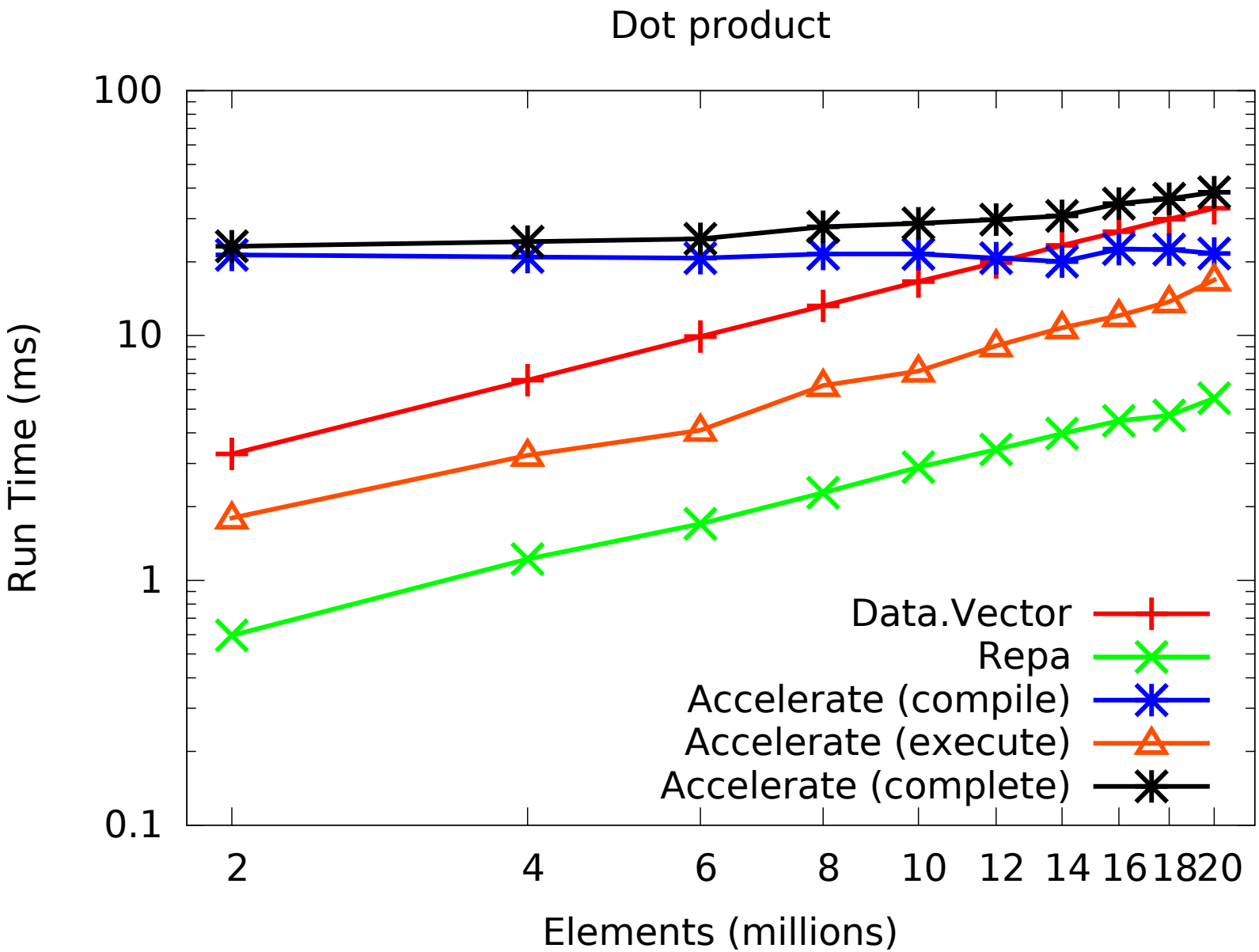
Local value numbering

```
writeVariable(variable, block, value):
    currentDef[variable][block] <- value

readVariable(variable, block):
    if currentDef[variable] contains block:
        # local value numbering
        return currentDef[variable][block]
    # global value numbering
    return readVariableRecursive(variable, block)
```

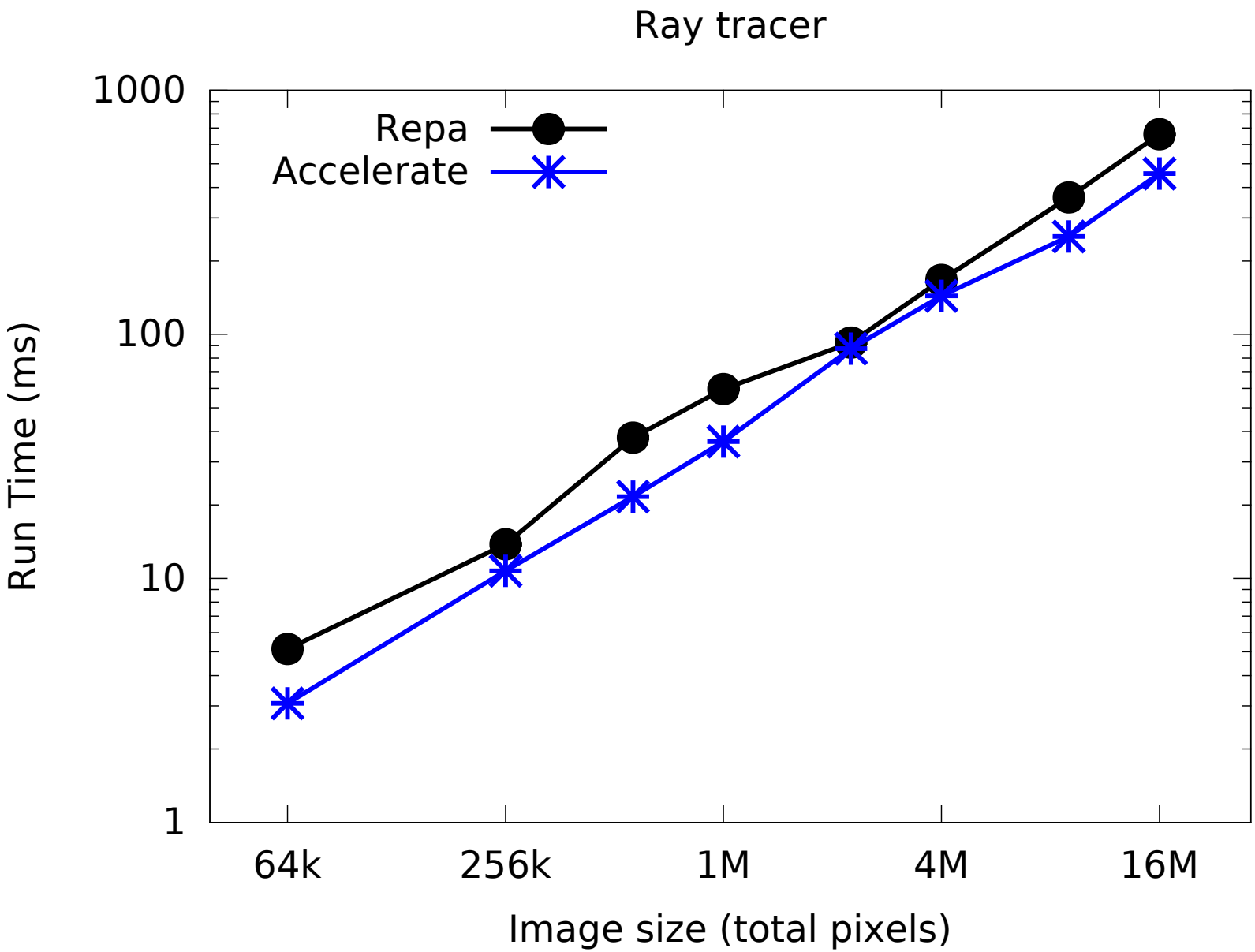
Global value numbering

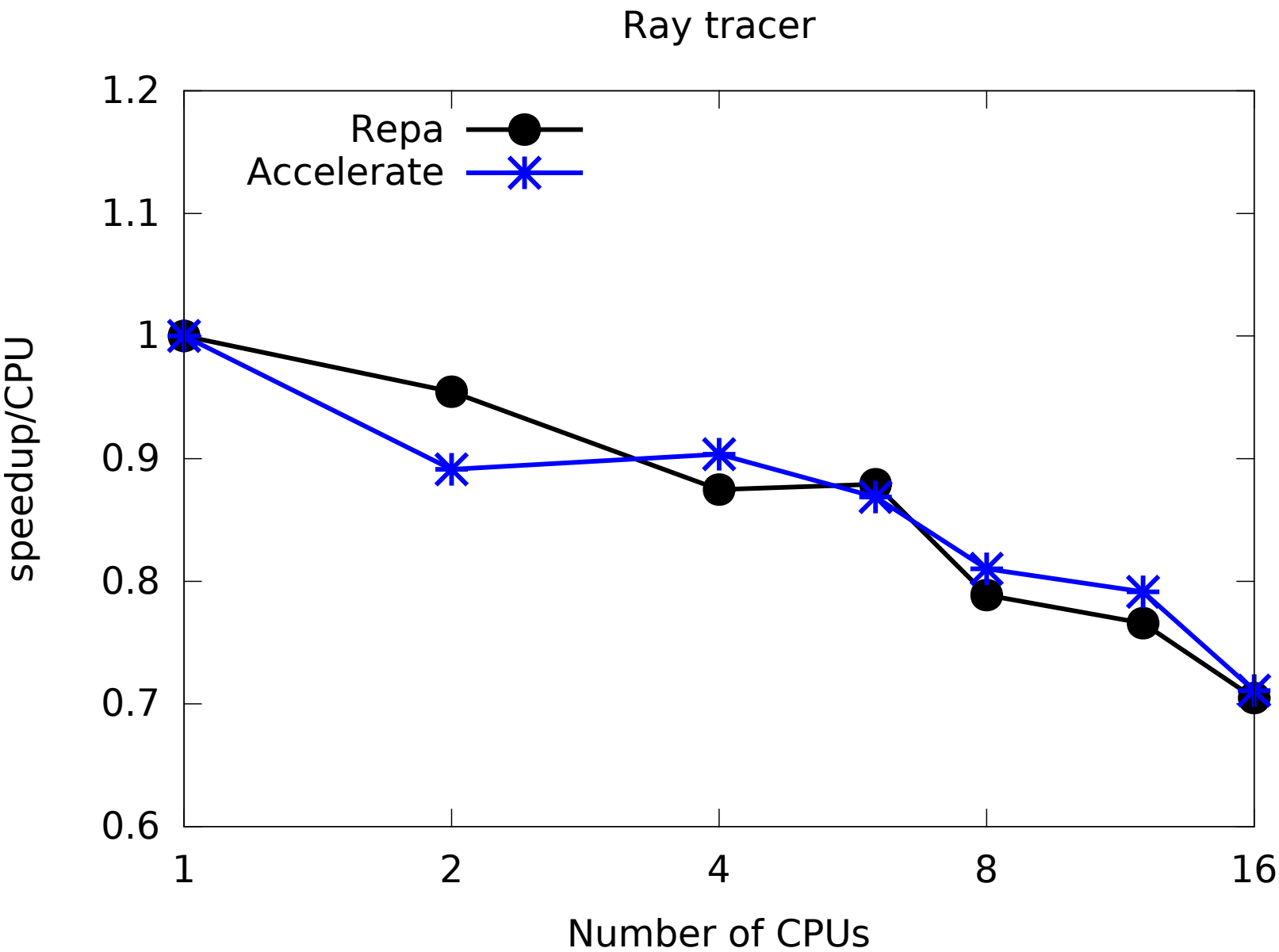
```
readVariableRecursive(variable, block):
    if |block.preds| = 0:
        # First block
        val <- variable
    else:
        # Break potential cycles with operandless phi
        val <- new Phi(block)
        writeVariable(variable, block, val)
    writeVariable(variable, block, val)
    return val
```

```
define i64 @foo(i64 %start, i64 %end) {
entry:
  br label %for.head

for.head:                                 ; preds = %n0, %entry
  %x.12 = phi i64 [ 0, %entry ], [ %x.6, %n0 ]
  %i.4 = phi i64 [ %start, %entry ], [ %i.9, %n0 ]
  %for.cond.3 = icmp slt i64 %i.4, %end
  br i1 %for.cond.3, label %n0, label %for.end

n0:                                       ; preds = %for.head
  %x.6 = add i64 %i.4, %x.12
  %i.9 = add nuw nsw i64 %i.4, 1
  br label %for.head

for.end:                                  ; preds = %for.head
  ret i64 %x.12
}
```


Dot product
Run Time (ms)
100
10
1
0.1
2
4
6
8
10
12
14
16
18
20
Elements (millions)
Data.Vector
Repa
Accelerate (compile)
Accelerate (execute)
Accelerate (complete)

Step	Time (ms)
getTarget	0.098
llvmOfAcc	0.0771
startFunction	0.0403
withContext	0.0879
withModuleFromAST	6.8032
withMachine	0.2997
libinfo	0.0801
optimizeModule	4.0379
withMCJIT	0.05
withModuleInEngine	0.2374
getGlobalFunctions	9.8508


Ray tracer
Repa
Accelerate
Run Time (ms)
1000
100
10
1
64k
256k
1M
4M
16M
Image size (total pixels)


Ray tracer
Repa
Accelerate
speedup/CPU
1.2
1.1
1
0.9
0.8
0.7
0.6
1
2
4
8
16
Number of CPUs