GVN-Hoist: Hoisting Computations from Branches

Sebastian Pop and Aditya Kumar

SARC: Samsung Austin R&D Center

November 3, 2016

CFGSimplify's code hoisting

- hoists computations at the beginning of BB
- stops at first difference
- very fast: disabling slows the compiler: 1688 $\rightarrow$ 1692 Bn insns

CFGSimplify limits

```
if (inv >= 0) {
  tmin = (min - a) * inv;
  tmax = (max - a) * inv;
} else {
  tmin = (max - a) * inv;
  tmax = (min - a) * inv;
}
```

```
x = (min - a) * inv;
y = (max - a) * inv;
if (inv >= 0) {
  tmin = x;
  tmax = y;
} else {
  tmin = y;
  tmax = x;
}
```

GVN-Hoist: Hoisting Computations from Branches

- identifies identical computations in a function
- hoist identical computations to a common dominator
- reduces code size
- reduces critical path length by exposing more ILP

Optimistic GVN-hoist Algorithm

- compute value number of scalars, loads, stores, calls
- compute insertion points of each type of instructions
- hoist expressions and propagate changes by updating SSA

Value Numbering

Simple program

```
a = x + y
b = x + 1
c = y + 1
d = b + c
e = a + 2
```

Value Numbering

Simple program

```
a = x + y
b = x + 1
c = y + 1
d = b + c
e = a + 2
```

$\rightarrow$

Value Numbering

```
a -> 1
b -> 2
c -> 3
d -> 4
e -> 4
```

GVN-Hoist: Algorithm-collecting value numbers

- scalars: use the existing GVN infrastructure
- loads: VN the pointer operand
- stores: VN the pointer operand and the value being stored
- calls: as stores, loads, or scalars (following side effects)

current GVN not accurate for loads and stores: use ad-hoc change

GVN-Hoist: Algorithm-compute insertion points

Insertion Point: A location where all the operands are either available or, can be made available.

- Compute a common insertion point for a set of instructions having the same GVN (Similar to VBEs but not as strict)
- Partition the candidates into a smaller set of hoistable candidates when no common insertion points can be found

GVN-Hoist: Algorithm-hoist expressions

- scalars: just move one of the instructions to the hoisting point and remove others; update SSA
- loads and stores: try to make geps available, then hoist; update SSA and memory SSA

Example

```
define float @f(float %d, float %min, float %max, float %a) {
entry:
  %div = fdiv float 1.000000e+00, %d
  %cmp = fcmp oge float %div, 0.000000e+00
  br i1 %cmp, label %if.then, label %if.else

if.then:
; preds = %entry
  %sub = fsub float %min, %a
  %mul = fmul float %sub, %div
  %sub1 = fsub float %max, %a
  %mul2 = fmul float %sub1, %div
  br label %if.end

if.else:
; preds = %entry
  %sub3 = fsub float %max, %a
  %mul4 = fmul float %sub3, %div
  %sub5 = fsub float %min, %a
  %mul6 = fmul float %sub5, %div
  br label %if.end

if.end:
; preds = %if.else, %if.then
  %tmax.0 = phi float [ %mul2, %if.then ], [ %mul6, %if.else ]
  %tmin.0 = phi float [ %mul, %if.then ], [ %mul4, %if.else ]
  %add = fadd float %tmax.0, %tmin.0
  ret float %add
}
```

Example

```
define float @f(float %d, float %min, float %max, float %a) {
entry:
  %div = fdiv float 1.000000e+00, %d
  %cmp = fcmp oge float %div, 0.000000e+00
  %sub1 = fsub float %max, %a
  %sub = fsub float %min, %a
  %mul2 = fmul float %sub1, %div
  %mul = fmul float %sub, %div
  br i1 %cmp, label %if.then, label %if.else

if.then:
; preds = %entry
  br label %if.end

if.else:
; preds = %entry
  br label %if.end

if.end:
; preds = %if.else, %if.then
  %tmax.0 = phi float [ %mul2, %if.then ], [ %mul, %if.else ]
  %tmin.0 = phi float [ %mul, %if.then ], [ %mul2, %if.else ]
  %add = fadd float %tmax.0, %tmin.0
  ret float %add
}
```

Cost models

- limit the number of basic blocks in the path between initial position and the hoisting point
- limit the number of instructions between the initial position and the beginning of its basic block
- do not hoist GEPs
- limit the number of dependent instructions to be hoisted

GVN hoisting

- 1% compile time overhead: 1678 $\rightarrow$ 1692 Bn insns
- more hoists than CFG-simplify: 15048 $\rightarrow$ 25318

Scalars hoisted	8960
Scalars removed	11940
Loads hoisted	16301
Loads removed	22690
Stores hoisted	50
Stores removed	50
Calls hoisted	7
Calls removed	7
Total Instructions hoisted	25318
Total Instructions removed	34687

Code size reduction

Code-size metric (.text)	Number
Total benchmarks	497
Total gained in size	39
Total decrease in size	58
Median decrease in size	2.9%
Median increase in size	2.4%