LLVM & HPSSA

Hot Path SSA Form in LLVM

Presented By Abhay[1] & Muzzammil[1]

[1]IIT Kanpur
PRAISE Group

Dr. Subhajit Roy, Dr. Awanish Pandey, Mr. Sumit Lahiri

What we modified in LLVM Source?

- New `llvm::intrinsic` signature, `"llvm.tau"` to support addition and removal of τ-functions to the LLVM SSA IR representation.

```
+ //===----------- intrinsic for tau ------------------=====//
+ def int_tau : DefaultAttrsIntrinsic<[llvm_any_ty],
+                                     [llvm_vararg_ty],
+                                     []>;
```

What we modified in LLVM Source?

- Modified Verifier::verifyDominatesUse() function since we don't want our intrinsic to interfere with dominators computation.

```
+ //===---------- Changes for tau.intrinsic ---------------=====//
void Verifier::verifyDominatesUse(Instruction &I, unsigned i) {
  Instruction *Op = cast<Instruction>(I.getOperand(i));
  + if (CallInst *CI = dyn_cast<CallInst>(&I)) {
  +   Function *CallFunction = CI->getCalledFunction();
  +   if (CallFunction != NULL && CallFunction->getIntrinsicID() ==
  +       Function::lookupIntrinsicID("llvm.tau")) {
  +     return;
  +   }
  + }
```

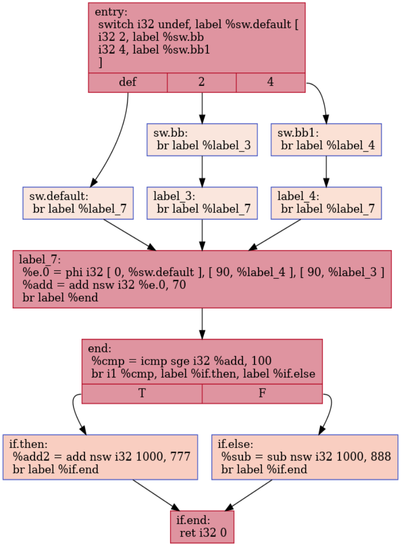
`HPSSAPass` : Overview

- Implemented `llvm::HPSSAPass` pass using the new LLVM Pass Manager.
- Pass `HPSSAPass::run(Function \&F, ...)` runs over a `llvm::Function` and inserts `"llvm.tau"` intrinsic calls with speculative and safe arguments at strategic positions in the LLVM IR as described in the previous slides.

Key HPSSA Data Structures :

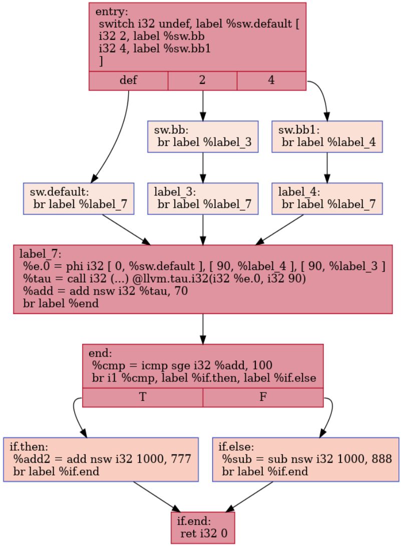
- Hot Path Set using `llvm::BitVector`.
- Definition Accumalator, `defAccumalate` as a map `std::map<{PHINode*,BasicBlock*}, {Value*,BitVector}>`.
- Variable Renaming Stack as a map, `std::map<Value*,Value*>`

HPSSA Transformation



```
entry:
 switch i32 undef, label %sw.default [
 i32 2, label %sw.bb
 i32 4, label %sw.bb1
 ]

sw.bb:
 br label %label_3

sw.bb1:
 br label %label_4

sw.default:
 br label %label_7

label_3:
 br label %label_7

label_4:
 br label %label_7

label_7:
 %e.0 = phi i32 [ 0, %sw.default ], [ 90, %label_4 ], [ 90, %label_3 ]
 %add = add nsw i32 %e.0, 70
 br label %end

end:
 %cmp = icmp sge i32 %add, 100
 br i1 %cmp, label %if.then, label %if.else

if.then:
 %add2 = add nsw i32 1000, 777
 br label %if.end

if.else:
 %sub = sub nsw i32 1000, 888
 br label %if.end

if.end:
 ret i32 0
```

CFG for 'main' function



```
entry:
 switch i32 undef, label %sw.default [
 i32 2, label %sw.bb
 i32 4, label %sw.bb1
 ]

sw.bb:
 br label %label_3

sw.bb1:
 br label %label_4

sw.default:
 br label %label_7

label_3:
 br label %label_7

label_4:
 br label %label_7

label_7:
 %e.0 = phi i32 [ 0, %sw.default ], [ 90, %label_4 ], [ 90, %label_3 ]
 %tau = call i32 (...) @llvm.tau.i32(i32 %e.0, i32 90)
 %add = add nsw i32 %tau, 70
 br label %end

end:
 %cmp = icmp sge i32 %add, 100
 br i1 %cmp, label %if.then, label %if.else

if.then:
 %add2 = add nsw i32 1000, 777
 br label %if.end

if.else:
 %sub = sub nsw i32 1000, 888
 br label %if.end

if.end:
 ret i32 0
```

CFG for 'main' function

HPSSAPass : Main & Destruction Pass

- `HPSSAPass::run(Function \&F, FunctionAnalysisManager \&AM)`
- `llvm::Function::RPOT()`.
- `llvm::successors()`.
- `llvm::DominatorTreeAnalysis` and `llvm::dominates()`.
- Replace use of `phi's` with `tau` variables using renaming stack.
- Out of HPSSA Form.

HPSSAPass : Auxilliary Functions

- `HPSSAPass::getProfileInfo(Function \&F)`
- `HPSSAPass::getCaloricConnector(Function \&F)`
- `HPSSAPass::Search(BasicBlock \&BB, DomTreeNode \&DTN)`

New Additions to SCCP Pass

- Modified the existing SCCP Pass to add in `SCCPInstVisitor::visitTauNode()` function similar to `SCCPInstVisitor::visitPHINode()`, which handles the special `"llvm.tau"` intrinsic instructions added for τ-functions.
- Added a new lattice element type `"spec_constant"` in `ValueLattice` class supporting operations on speculative constants.
- Added new functions in the `SCCPInstVisitor` and `SCCPSolver` class to handle operations on speculative constants using `markSpeculativeConstant()` function.

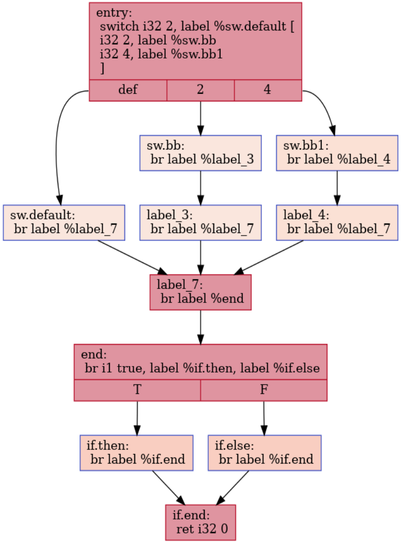
Further Modifications

- Modified the `SCCPInstVisitor::mergeIn()` function to handle lattice "meet" operation for the new speculative constants introduced.
- Since we added the τ-functions as an `"llvm.tau"` intrinsic which is essentially an `llvm:CallInst` type, we modified all appropriate visit and marking functions in `SCCPInstVisitor`, `SCCPSolver` and `SCCPPass` to handle this case separately by calling `visitTauNode()`.
- Modified utility functions in `SCCPInstVisitor` and `SCCPSolver` class to print marking of speculative constants and related operations for debugging purpose.

```
...
[BBWorkList] Visiting LLVM Instrinsic : llvm.tau (call)
Visiting Tau Instruction
Speculative Operand : , speculative constant
Speculative Operand : llvm.tau.i32, speculative constant
Merged speculative constant into   %tau = call i32 (...)
        @llvm.tau.i32(i32 %e.0, i32 90) : speculative constant
ValueLattice (TauState) : speculative constant
```

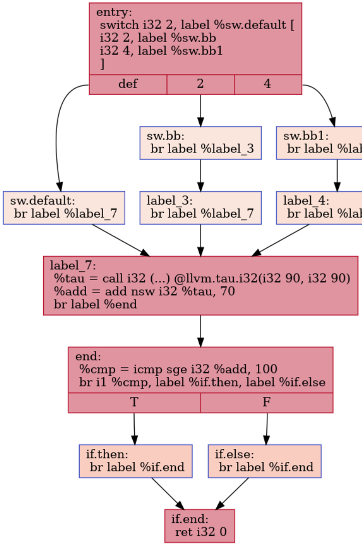
```
int main() {
        int a = 1000, z, c, e = 0;
        switch(c) {
                case 2 : goto label_3; break;
                case 4 : goto label_4; break;
                default : goto label_7;
        }
        label_3:
                e = 90;
                goto label_7;
        label_4:
                e = 100 - 10;
                goto label_7;
        label_7:
                e = e + 70;  // e in rhs is 90.
                goto end;
        end:
                if (e >= 100) {  // e is greater than 100 always
                        a = a + 777;
                } else {
                        a = a - 888;
                }
        return 0;
```

SSCCP with an Example



```
entry:
 switch i32 2, label %sw.default [
 i32 2, label %sw.bb
 i32 4, label %sw.bb1
 ]
def | 2 | 4

sw.bb:
 br label %label_3

sw.bb1:
 br label %label_4

sw.default:
 br label %label_7

label_3:
 br label %label_7

label_4:
 br label %label_7

label_7:
 br label %end

end:
 br i1 true, label %if.then, label %if.else
T | F

if.then:
 br label %if.end

if.else:
 br label %if.end

if.end:
 ret i32 0
```

CFG for 'main' function



```
entry:
 switch i32 2, label %sw.default [
 i32 2, label %sw.bb
 i32 4, label %sw.bb1
 ]
def | 2 | 4

sw.bb:
 br label %label_3

sw.bb1:
 br label %label_4

sw.default:
 br label %label_7

label_3:
 br label %label_7

label_4:
 br label %label_7

label_7:
 %tau = call i32 (...) @llvm.tau.i32(i32 90, i32 90)
 %add = add nsw i32 %tau, 70
 br label %end

end:
 %cmp = icmp sge i32 %add, 100
 br i1 %cmp, label %if.then, label %if.else
T | F

if.then:
 br label %if.end

if.else:
 br label %if.end

if.end:
 ret i32 0
```

CFG for 'main' function