UMBRA

Pleasant Debugging of Generated Code

Philipp Fent

fent@in.tum.de

Debugging generated code is hard

```
41:
  %7154 = phi i64 [%7094, %39 %8254, %43]
  %7194 = load int32 %7058, %7154
  %7216 = zext i64 %7194
  %7226 = crc32 i64 569013765, %7216
  %7240 = mul i64 %7226, -8770518540659720191
  %7254 = getelementptr int8 %state, i64 544
  %7276 = getelementptr object AggregationTarget %7254, i32 0, i32 0, i32 0
  %7320 = and i64 %7240, 511
  %7334 = getelementptr object AggregationTarget::Fragment %7276, %7320
  %7356 = load object AggregationTarget::Fragment %7334, i32 0, i32 0
  %7382 = isnull ptr %7356
  condbr %7382 %44 %47

47:
  %7410 = load object AggregationTarget::Fragment %7334, i32 0, i32 1
  %7436 = lshr i64 %7240, %7410
  %7450 = load object Preaggregation::EntryHeader* %7356, %7436
  %7472 = isnull ptr %7450
  condbr %7472 %44 %48

48:
  %7500 = phi ptr [%7450, %47 %7526, %49]
  %7580 = getelementptr int8 %7500, i64 20
  %7602 = load int8 %7580
  %7620 = testbit i8 %7602, 0
  %7634 = load int32 %7500, i32 4
  %7656 = cmpeq i32 %7634, %7194
  %7670 = not bool %7620
  %7680 = and bool %7656, %7670
  condbr %7680 %50 %49
```

What can we do about it?

```
#include <source_location> // C++20

struct CodeGenLocation {
   std::source_location debugLocation;
   CodeGenLocation(std::source_location l =
                   std::source_location::current());
   std::string getDebugName() const;
};

UInt64 add(UInt64 a, UInt64 b, CodeGenLocation loc = {});
// Similarly: basic blocks, function names, etc...
```

What can we do about it?

Before:

```
43:
  %8316 = add i64 %7154, 1
  %8330 = cmpne i64 %8316, %7156
  condbr %8330 %41 %42
```

After:

```
contScan_TableScanTranslator_cpp_462_:
  %RelationMappedLogic_cpp_667_ = add i64 %localTid, 1
  %12383 = cmpne i64 %RelationMappedLogic_cpp_667_, %RelationMappedLogic_cpp_657_3
  condbr %12383 %loopTuples_RelationMappedLogic_cpp_663_ %loopDoneTuples_RelationMappedLogic_cpp_663_
```

Slight problems

```
Int64 sum (Int64 a, Int64 b) {
   return a + b;
}

struct Int64 {
   // No debug info :(
   Int64 operator+(Int64 v) const;

   // Error: operator+() const' must have either zero or one arguments
   Int64 operator+(Int64 v, CodeGenLocation = {}) const;

   // Cute hack
   Int64 operator+(Wrapper<Int64> v) const;
}
```

Cute Hack

```
template <class T>
struct Wrapper : CodeGenLocation {
   T value;
   // Implicit conversion
   Wrapper(T value, std::source_location debugLocation =
                    std::source_location::current());
}

// Error: operator+ with 'size_t' is ambiguos :(
// candidates:
// operator+(Wrapper<int32_t>)
// operator+(Wrapper<uint32_t>)
// operator+(Wrapper<int64_t>)
// operator+(Wrapper<uint64_t>)
```

Cute Hack

```
template <class T>
struct Wrapper : CodeGenLocation {
   T value;
   // Implicit conversion (v2)
   template <class T2> requires(FittingType<T, T2>)
   Wrapper(T2&& value, std::source_location debugLocation =
                       std::source_location::current());
}

template <class T, class T2>
concept FittingType = std::is_convertible_v<T2*, T*>;
```

Result

```
loopTuples_RelationMappedLogic_cpp_663_:
  %localTid = phi i64 [%RelationMappedLogic_cpp_657_, %loopBlocks_RelationMappedLogic_cpp_649_ %RelationMappedLogic_cpp_667_, %contScan_TableSca
  %c_custkey = load int32 %RelationMappedLogic_cpp_655_2, %localTid
  %Hash_cpp_177_ = zext i64 %c_custkey
  %Hash_cpp_69_ = crc32 i64 569013765, %Hash_cpp_177_
  %Hash_hpp_286_ = mul i64 %Hash_cpp_69_, -8770518540659720191
  %EagerRightGroupJoinTranslator_cpp_269_ = getelementptr int8 %state, i64 544
  %fragments = getelementptr object umbra::AggregationTarget %EagerRightGroupJoinTranslator_cpp_269_, i32 0, i32 0, i32 0
  %PreaggregationLogic_cpp_199_ = and i64 %Hash_hpp_286_, 511
  %PreaggregationLogic_cpp_199_4 = getelementptr object umbra::AggregationTarget::Fragment %fragments, %PreaggregationLogic_cpp_199_
  %hashTable5 = load object umbra::AggregationTarget::Fragment %PreaggregationLogic_cpp_199_4, i32 0, i32 0
  %PreaggregationLogic_cpp_204_ = isnull ptr %hashTable5
  condbr %PreaggregationLogic_cpp_204_ %emptyRight_EagerRightGroupJoinTranslator_cpp_263_ %cont_PreaggregationLogic_cpp_204_

cont_PreaggregationLogic_cpp_204_:
  %hashTableShift = load object umbra::AggregationTarget::Fragment %PreaggregationLogic_cpp_199_4, i32 0, i32 1
  %PreaggregationLogic_cpp_205_ = lshr i64 %Hash_hpp_286_, %hashTableShift
  %PreaggregationLogic_cpp_205_6 = load object umbra::Preaggregation::EntryHeader* %hashTable5, %PreaggregationLogic_cpp_205_
  %PreaggregationLogic_cpp_210_ = isnull ptr %PreaggregationLogic_cpp_205_6
  condbr %PreaggregationLogic_cpp_210_ %emptyRight_EagerRightGroupJoinTranslator_cpp_263_ %loopChain_PreaggregationLogic_cpp_209_

loopChain_PreaggregationLogic_cpp_209_:
  %iter = phi ptr [%PreaggregationLogic_cpp_205_6, %cont_PreaggregationLogic_cpp_204_ %next, %notEqual_PreaggregationLogic_cpp_219_]
  %MaterializationHelper_cpp_713_ = getelementptr int8 %iter, i64 20
  %MaterializationHelper_cpp_714_ = load int8 %MaterializationHelper_cpp_713_
  %MaterializationHelper_cpp_770_ = testbit i8 %MaterializationHelper_cpp_714_, 0
  %MaterializationHelper_cpp_849_ = load int32 %iter, i32 4
  %11349 = cmpeq i32 %MaterializationHelper_cpp_849_, %c_custkey
  %TypeLogic_cpp_325_ = not bool %MaterializationHelper_cpp_770_
  %TypeLogic_cpp_325_7 = and bool %11349, %TypeLogic_cpp_325_
  condbr %TypeLogic_cpp_325_7 %cont_ConsumerContext_cpp_455_ %notEqual_PreaggregationLogic_cpp_219_
```