Compilation of Cursor Loops by Realizing Aggify "Correctly"

Haoyu Zhang, Yuchen Liu

Carnegie Mellon University, Course 15745 Project: Poster Session

INTRODUCTION & MOTIVATION

Databases support User Defined Functions
Oracle, IBM DB2, PostgreSQL, and SQL Server

UDF Inlining speedups User Defined Function by A LOT

- Froid, Apfel increase UDF performance to near pure SQL level
- 1000x faster than before

Aggify can rewrite <u>Cursor Loops</u> to Custom Aggregates

- The only case that UDF Inlining cannot handle
- Inlining works for all UDFs now

NO ONE HAS IMPLEMENTED IT CORRECTLY UNTIL US

PROPOSED METHODS/DESCRIPTION

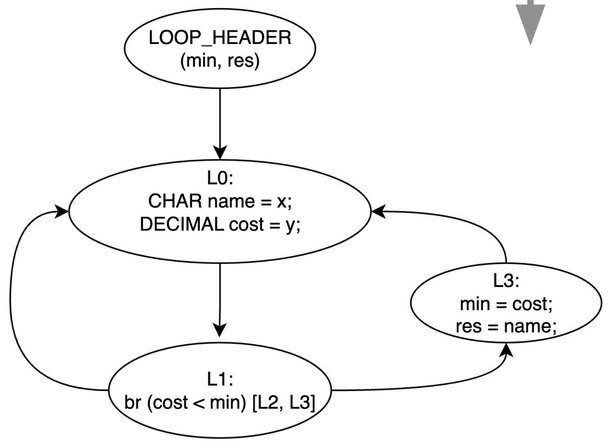
Construct CFG for PL/pgSQL
SQL Source => JSON => CFG
Perform Data Flow Analysis on the CFG
Liveness, Reaching Definitions, Use-Def Chain
Compile the loop body's CFG into a Custom Aggregate in C++ {init(), update(), finalize()}

- Compilation of Basic Blocks → goto
- Compilation of Instructions → Futamura projection on Logical Plan

Rewrite the Cursor Loop
Original PL/pgSQL with cursor loop => An equivalent query that invokes the custom aggregate

```
val := ?
FOR fetchedcost, fetchedname IN
(
        SELECT ps_supplycost,
               s_name
        FROM   partsupp,
               supplier
        WHERE  ps_partkey = KEY
        AND    ps_suppkey = s_suppkey)
LOOP
  IF fetchedcost < mincost THEN
    mincost := fetchedcost;
    val := fetchedname;
  END IF;
END LOOP;
```

PL/pgSQL




```
val :=
SELECT
   CASE
      WHEN count(*) > 0
   THEN custom_agg
     (fetchedcost,
         fetchedname,
         val)
   ELSE val
   END
FROM (...);
```

PL/pgSQL

```
if (LessThan::Operation(
CastLongtoDecimal(fetchedcost, 15, 2),
mincost)){
   mincost = fetchedcost;
   val = fetchedname;
}
```

C++

RESULTS