DAPHNE

Loops of the Domain-specific Programming Language DaphneDSL

Borko Bošković, Janez Brest, Aleš Zamuda

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 957407.

DAPHNE

- DAPHNE - integrated Data Analysis Pipelines for large-scale data management, Highperformance computing, and machiNE learning
- Open source: https://github.com/daphne-eu/daphne/
- DaphneDSL
  - variables, data types,
  - comments, expressions,
  - control structures,
  - loops, function, etc.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement number 957407.

DAPHNE

System Architecture

- Execution environment – the flows and operations as defined in the DaphneDSL
- Multi-level translation
- Multi-Level Intermediate Representation MLIR
  - Extensible compilers
  - Fragmented software
  - Compiling for heterogeneous hardware
  - Linking compilers
- DaphneDSL scripts => DaphneIR intermediate (MLIR)
- Efficient pipeline design and kernel execution

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement number 957407.

DAPHNE

Program for calculating the sum of the first 1000 natural numbers.

```
# File:  program.daphne
# Initialization
sum = 0;

# Calculation of sum
for (i in 1:1000) {
   sum = sum + i;
}

# Output result
print("Sum is:  "+sum);
```

Parsing, compiling and running the program.

```
$ bin/daphne program.daphne
$ Sum is:  500500
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 957407.

DAPHNE

DaphneIR dialect

```
module {
func.func @main() {
 %0="daphne.constant"(){value = 0 :  si64}:() -> si64
 %1="daphne.constant"(){value = "Sum is:  "}:()->!daphne.String
 ...
 %11=scf.for %arg0=%6 to %5 step %6 iter_args(%arg1=%0)->(si64){
    %c1_i32=arith.constant 1:i32
    %14="daphne.call_kernel"(%arg0,%c1_i32,%10){callee=
     "_cast__int64_t__size_t"}:(index,i32,!daphne.DaphneContext)->si64
    %c2_i32=arith.constant 2:i32
    %15="daphne.call_kernel"(%14,%4,%c2_i32,%10){callee=
     "_ewMul__int64_t__int64_t__int64_t"}:
     (si64,si64,i32,!daphne.DaphneContext)->si64
   %c3_i32=arith.constant 3:i32
   %16="daphne.call_kernel"(%arg1,%15,%c3_i32,%10){callee=
    "_ewAdd__int64_t__int64_t__int64_t"}:
    (si64,si64,i32,!daphne.DaphneContext)->si64
   scf.yield %16:si64
}
...
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement number 957407.

DAPHNE

Low-Autocorrelation Binary Sequences

$$Z_L = \{z_1, z_2, ..., z_L\};\ z_i \in \{+1, -1\}$$

$$E(Z_L) = \sum_{k=1}^{L-1} C_k^2$$

$$C_k(Z_L) = \sum_{i=1}^{L-k} z_i \cdot z_{i+k}$$

$$Z_L^* = \arg\min_{Z_L \in B_L} E(Z_L)$$

```
def calcE(s:matrix<si64>,L:ui64) ->
ui64 {
   E=as.ui64(0);
   for(k in 1:L - 1) {
      ck=as.si64(0);
      for(i in 0:L - k - 1) {
         ck=ck+as.si64(
         as.scalar(s[i,]*s[i+k,]));
      }
   tmp = as.ui64(ck*ck);
   E = E + tmp;
   }
   return E;
}

for (L in 5:301) {
   s = sample(2,L,true,-1);
   s = 2*s - 1;
   E = calcE(s,as.ui64(L));
   print("L="+L+" E="+E);
}
```

This project has received funding from the European Union’s Horizon 2020
research and innovation programme under grant agreement number 957407.

DAPHNE

Loops in the DaphneDSL

- A simple syntax
- Similar to programming languages, such as C++ and R
- Shorter and more expressive programs
- Larger number of iterations
  - Summarize the natural numbers up to the number 174,355
  - Calculated the energies for sequences of lengths up to L = 192

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement number 957407.

Stack size used by the program

- $ ulimit -s unlimited
- Summarize the natural numbers up to the number 10,000,000
- Calculated the energies for sequences of lengths up to L = 1000
- Developer quickly fixed the issue #77
- Open source code

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement number 957407.

DAPHNE

Conclusion

- DAPHNE project
- System Architecture
- Two examples of using DaphneDSL
- Loops in the DaphneDSL
- Quickly fixed the issue #77 (open source code)
- The DAPHNE code is more and more reliable and valuable over time

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement number 957407.