University
POLITEHNICA
of Bucharest

Faculty of
Automatic Control
and Computers

Computer Science
and Engineering
Department

Replacing DRuntime Hooks with Templates Across Three SAoCs

17 September 2024

Teodor Duțu (Teo)
teodor.dutu@gmail.com

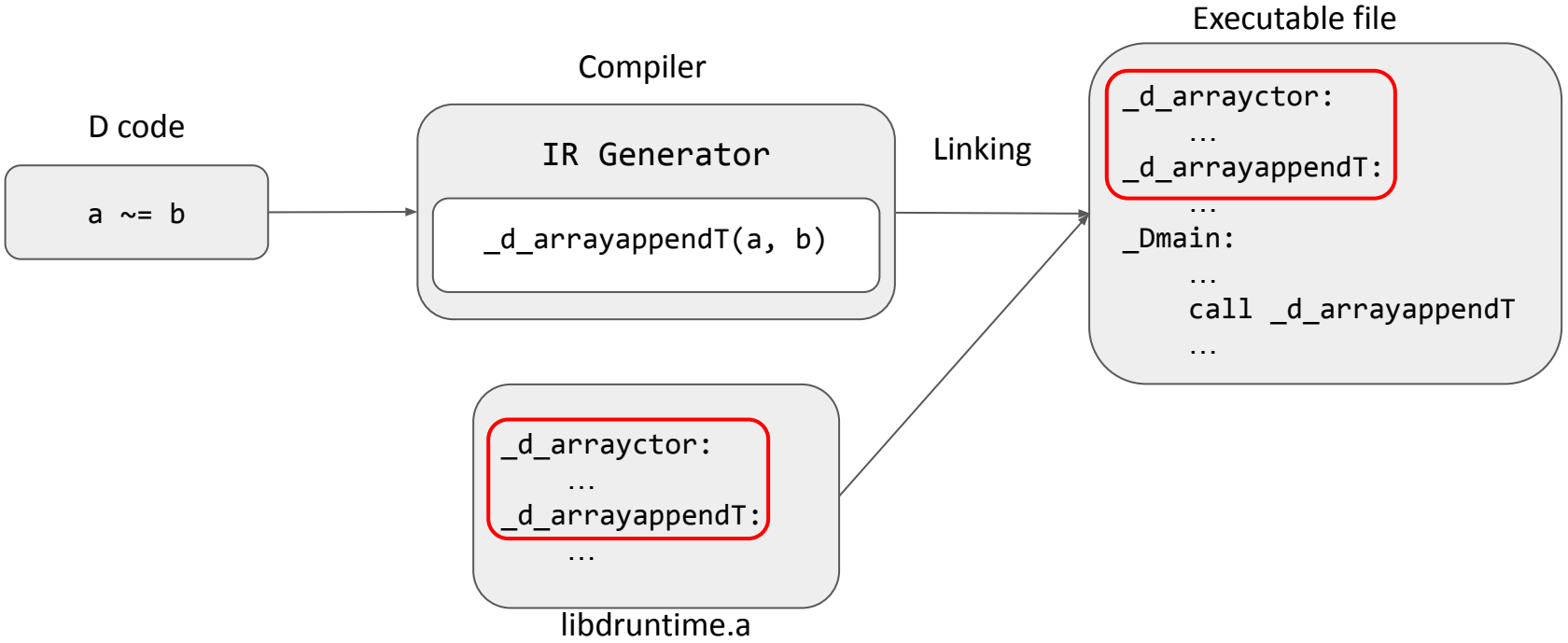
Răzvan Nițu

DRuntime - What?

- `string[string] map;`
- `arr[0 .. i - 1] ~ arr[i + 1 .. $];`
- `throw new Exception("I use DRuntime, bro!");`
- `new Class();`
- `int[] arr = [1, 2, 3];`

`-betterC`

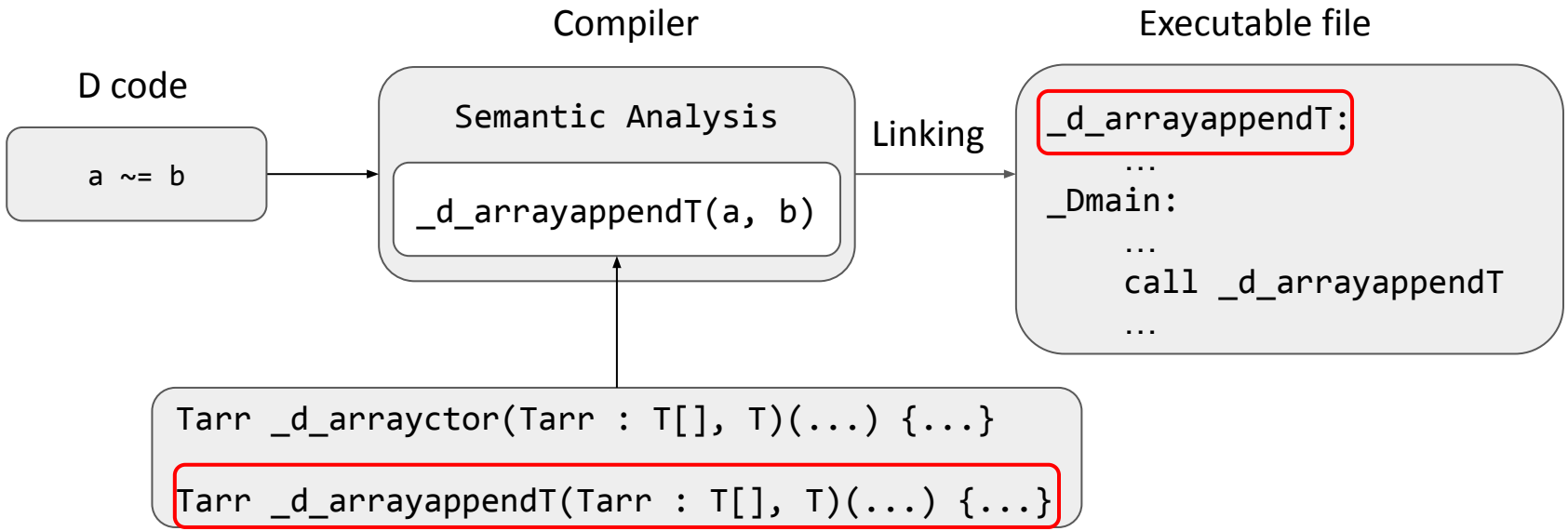
DRuntime - How (High Level)?



DRuntime - How? Non-Template Hook

```
extern (C) void[] _d_arrayappendT(
    const TypeInfo ti, ref byte[] x, byte[] y) @weak
{
    import core.stdc.string;
    auto length = x.length;
    auto tinext = unqualify(ti.next);
    auto sizeelem = tinext.tsize;

    _d_arrayappendcTX(ti, x, y.length);

    memcpy(x.ptr + length * sizeelem, y.ptr, y.length * sizeelem);
    __doPostblit(x.ptr + length * sizeelem, y.length * sizeelem, tinext);

    return x;
}
```

DRuntime - New How (High Level)?



DRuntime - How? Template Hook

```
ref Tarr _d_arrayappendT(Tarr : T[], T)(return ref scope Tarr x, scope Tarr y)
{
    _d_arrayappendcTX(x, y.length);

    static if (hasElaborateCopyConstructor!T && !hasPostblit!T)
        foreach (i, ref elem; y)
            copyEmplace(elem, x[x.length + i]);
    else if (y.length)
    {
        auto xptr = cast(Unqual!T *)&x[length];
        memcpy(xptr, cast(Unqual!T *)&y[0], y.length * T.sizeof);
        static if (hasPostblit!T)
            for (auto ptr = xptr; ptr < xptr + y.length; ptr++)
                ptr.__xpostblit();
    }

    return x;
}
```

CommaExps: Good-ish

```
S[] arr1, arr2;

arr1 ~= arr2;
// lowering:
_d_arrayappendT(arr1, arr2);

arr1 ~= S();
// lowering:
_d_arrayappendcTX(arr1, 1), arr1[$ - 1] = S(), arr1;
```

CommaExps: Meh

```
int cnt;
S[] arr;

S[] foo()
{
    cnt++;
    return arr;
}

foo() ~= S();
assert(cnt == 1);

// lowering:
_d_arrayappendcTX(foo(), 1), foo()[$ - 1] = S(), foo();
assert(cnt == 1);
```

CommaExps: Meh - SAoC 2021

```
int cnt;
S[] arr;

S[] foo()
{
    cnt++;
    return arr;
}

foo() ~= S();
assert(cnt == 1);

// lowering:
S[] _tmp = foo(), _d_arrayappendcTX(_tmp, 1), _tmp[$ - 1] = S(), _tmp;
assert(cnt == 1);
```

CommaExps: Oh Ok…

```d
int val = 2;

auto noEvil()
{
    class C
    {
        int* p;
        this() { p = &val; }
    }
    return new C();
}

void main()
{
    auto c = noEvil();
    *c.p = 3;  // val = 3
}
```

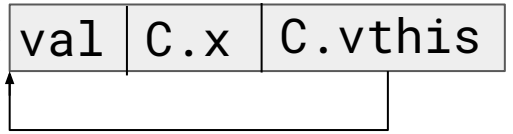
lowering

```d
return (
_tmp = _d_newclass!C(),
_tmp.__ctor());
```

CommaExps: Oh No…

```
auto evil()
{
    int val = 2;
    class C
    {
        int* p;
        this() { p = &val; }
    }
    return new C();
}

void main()
{
    auto c = evil();
    *c.p = 3;  // val = 3
}
```

```
class C
{
  int x;
  void* vthis;
}
```



closure

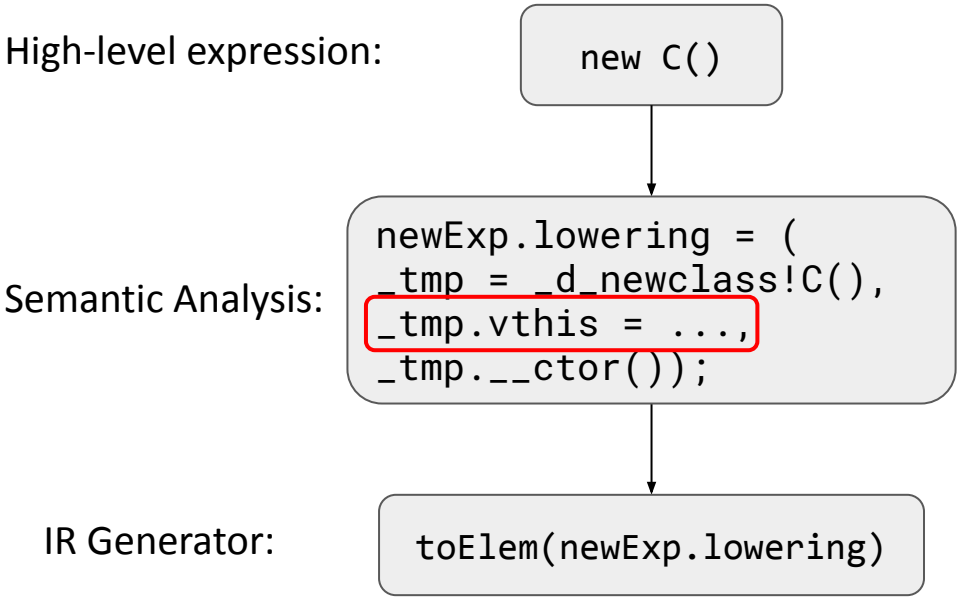
CommaExps: Oh No…

```
auto evil()
{
    int val = 2;
    class C
    {
        int* p;
        this() { p = &val; }
    }
    return new C();
}

void main()
{
    auto c = evil();
    *c.p = 3;  // val = 3
}
```

lowering

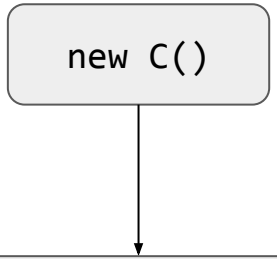
```
return (
_tmp = _d_newclass!C(),
_tmp.vthis = ...,
_tmp.__ctor());
```

What to Lower When? (SAoC 2021)



What to Lower When? Previously

High-level expression:



IR Generator:

```
ex = _d_newclass(TypeInfo_Class(C));
ey = ...;  // copy vthis
ez = __ctor();
el_combine(ex, ey, ez);
```

Split Lowering (SAoC 2022)

High-level expression:

```
new C()
```

Semantic Analysis:

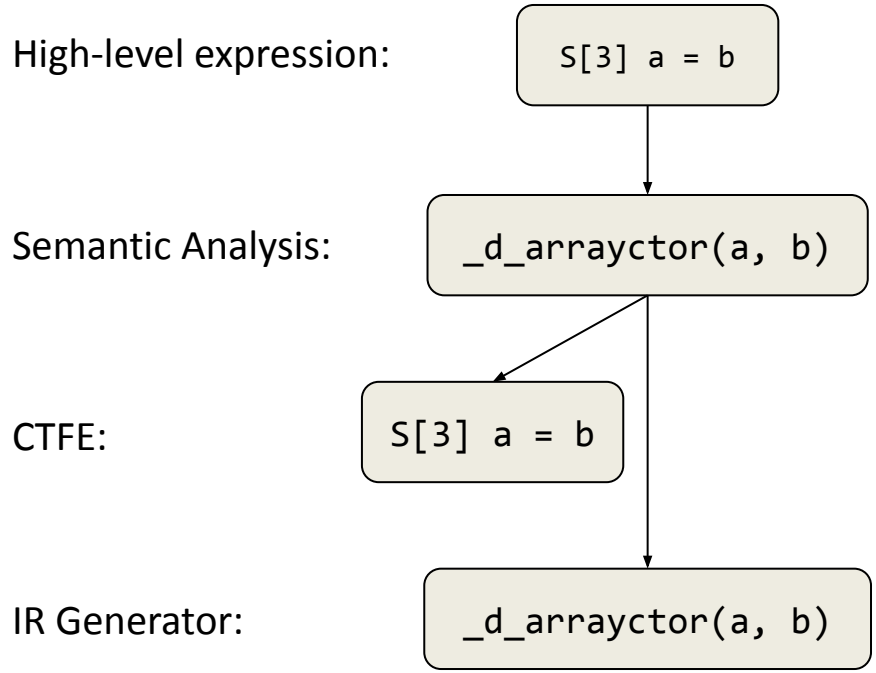
```
newExp.lowering = _d_newclass!C();
```

IR Generator:

```
ex = toElem(newExp.lowering);
ey = ...;  // copy vthis
ez = __ctor();
el_combine(ex, ey, ez);
```

What to Interpret? SAoC 2021

```
struct S { this(this) { } }

S[3] foo(S[3] b)
{
    S[3] a = b;
    return a;
}
static assert(foo(b));
```



What to Interpret? SAoC 2022

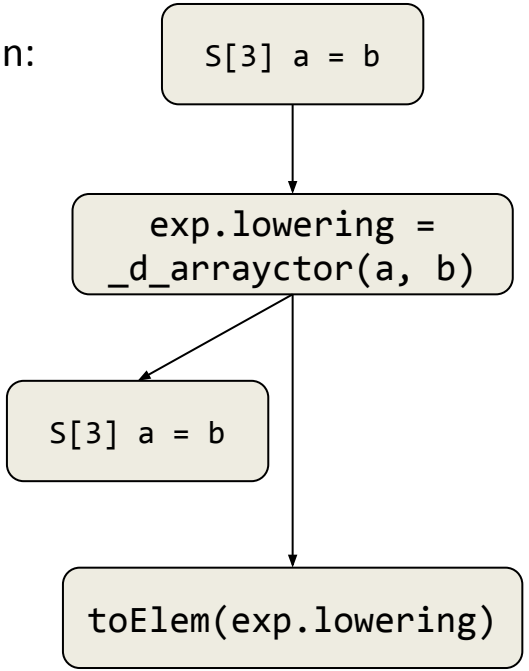
```
struct S { this(this) { } }

S[3] foo(S[3] b)
{
    S[3] a = b;
    return a;
}
static assert(foo(b));
```

High-level expression:

Semantic Analysis:

CTFE:

IR Generator:



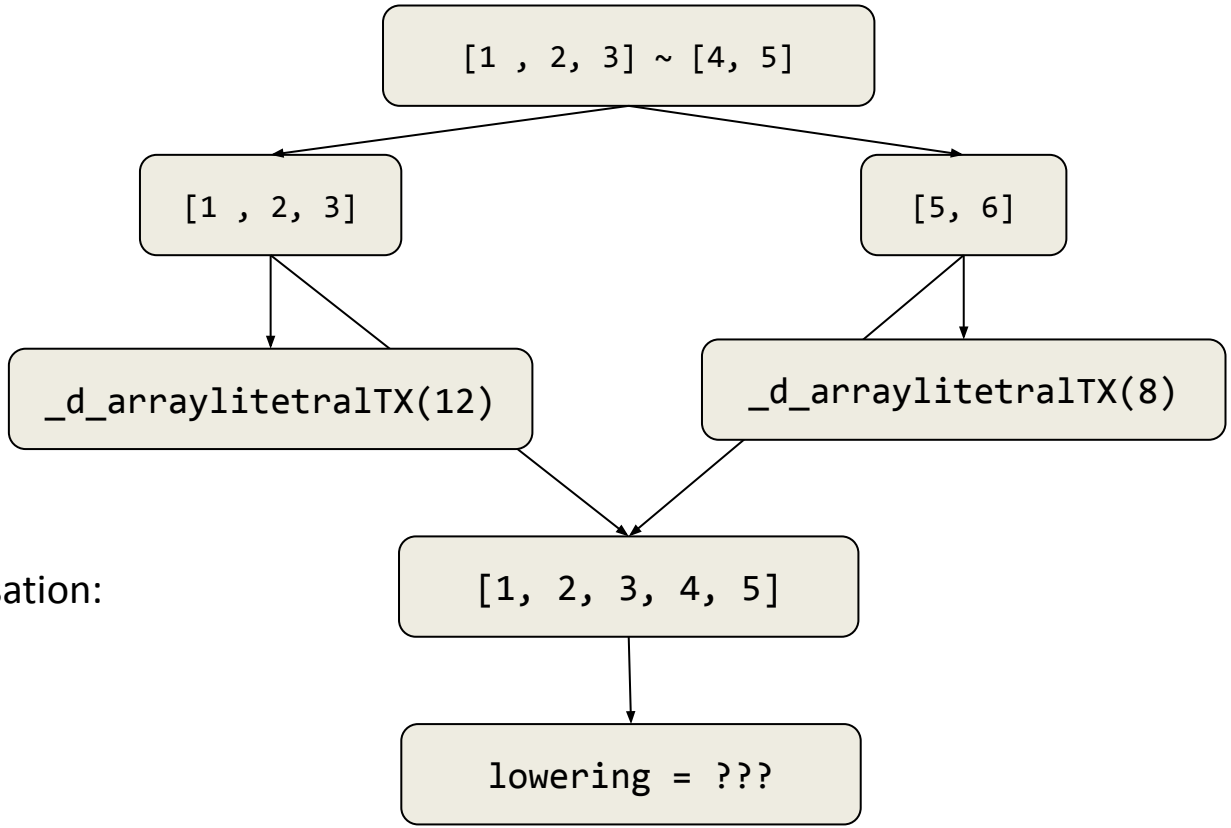
DMD Architecture (SAoC 202{1,2})

High-level expression:

Semantic Analysis:

Const Folding + Optimisation:

IR Generator:



DMD Architecture (SAoC 2023 - TODO)

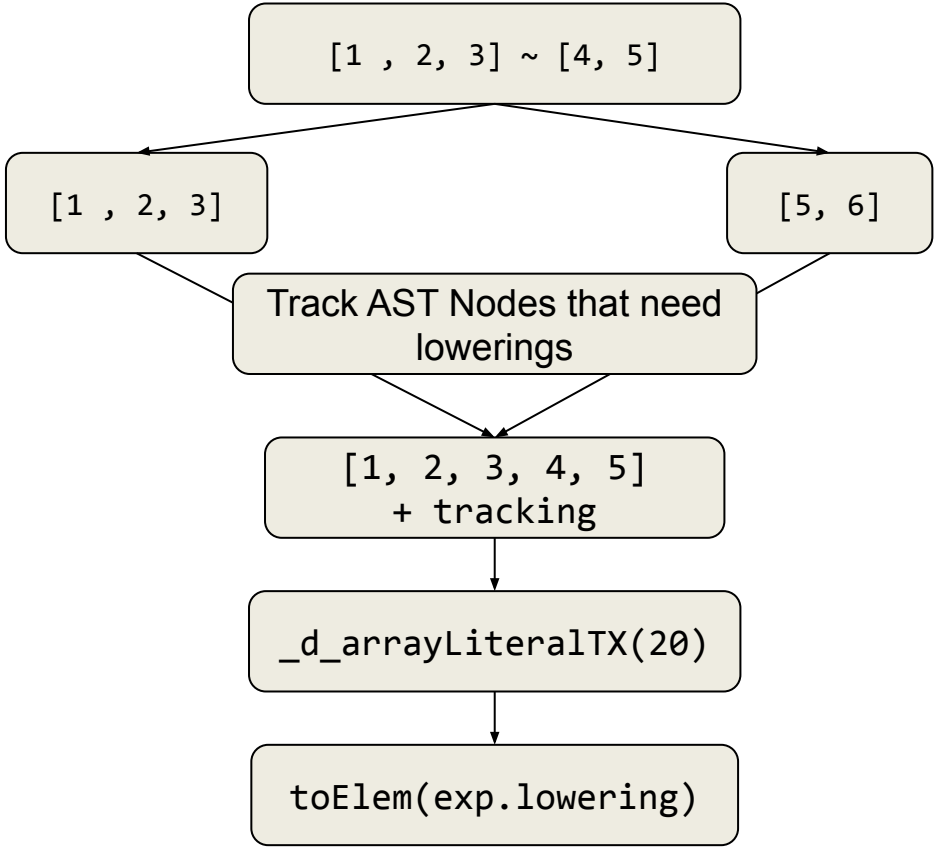
High-level expression:

Semantic Analysis:

Const Folding + Optimisation:

Lowering Compiler Pass:

IR Generator:



DRuntime Hooks Status

_d_arrayctor
_d_arraysetctor
_d_arrayassign
_d_arrayassign_l
_d_arrayassign_r
_d_delstruct
_d_newThrowable
_d_arrayappendT
_d_arrayappendcTX

SAoC 2021

_d_arraycatT
_d_arraycatnTX
_d_newitemiT
_d_newitemT
_d_newitemU
_d_newclass

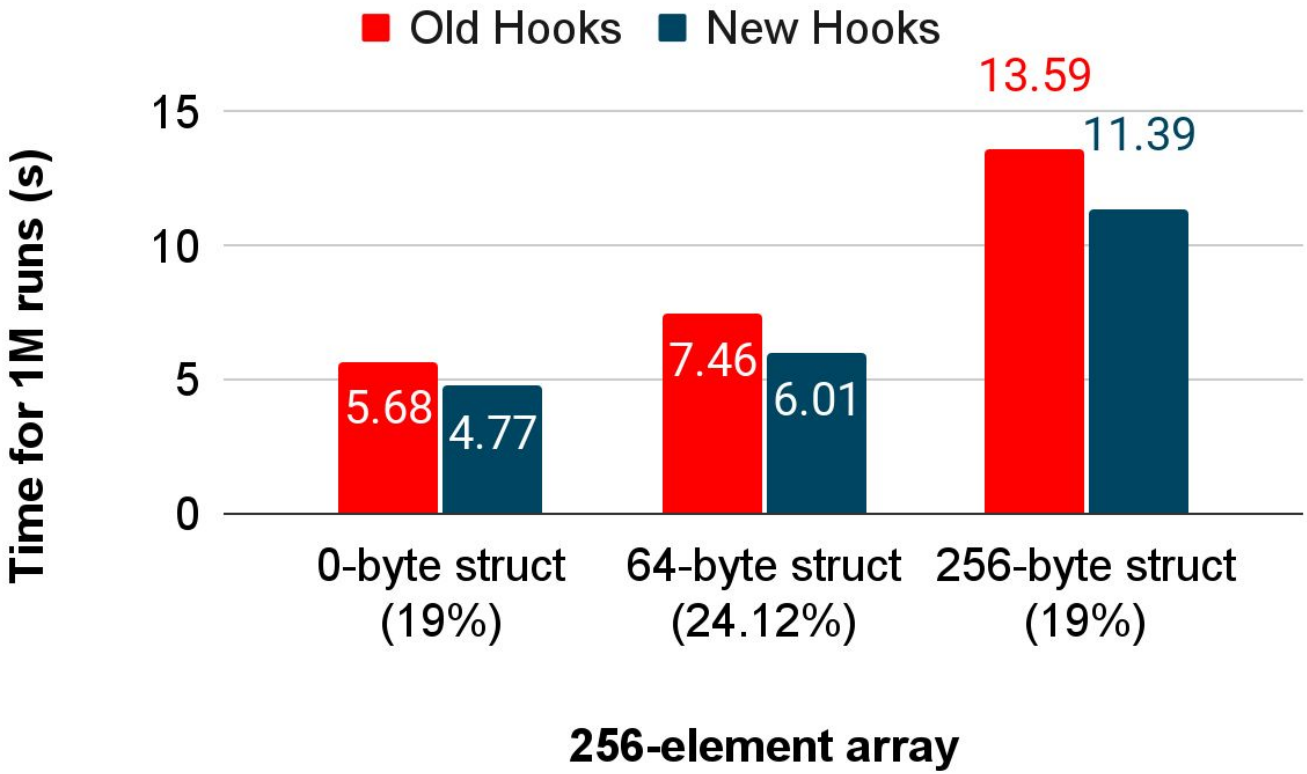
SAoC 2022

_d_newarrayiT
_d_newarrayT
_d_newarrayU
_d_newarraymTX
_d_newarrayOpT

SAoC 2023

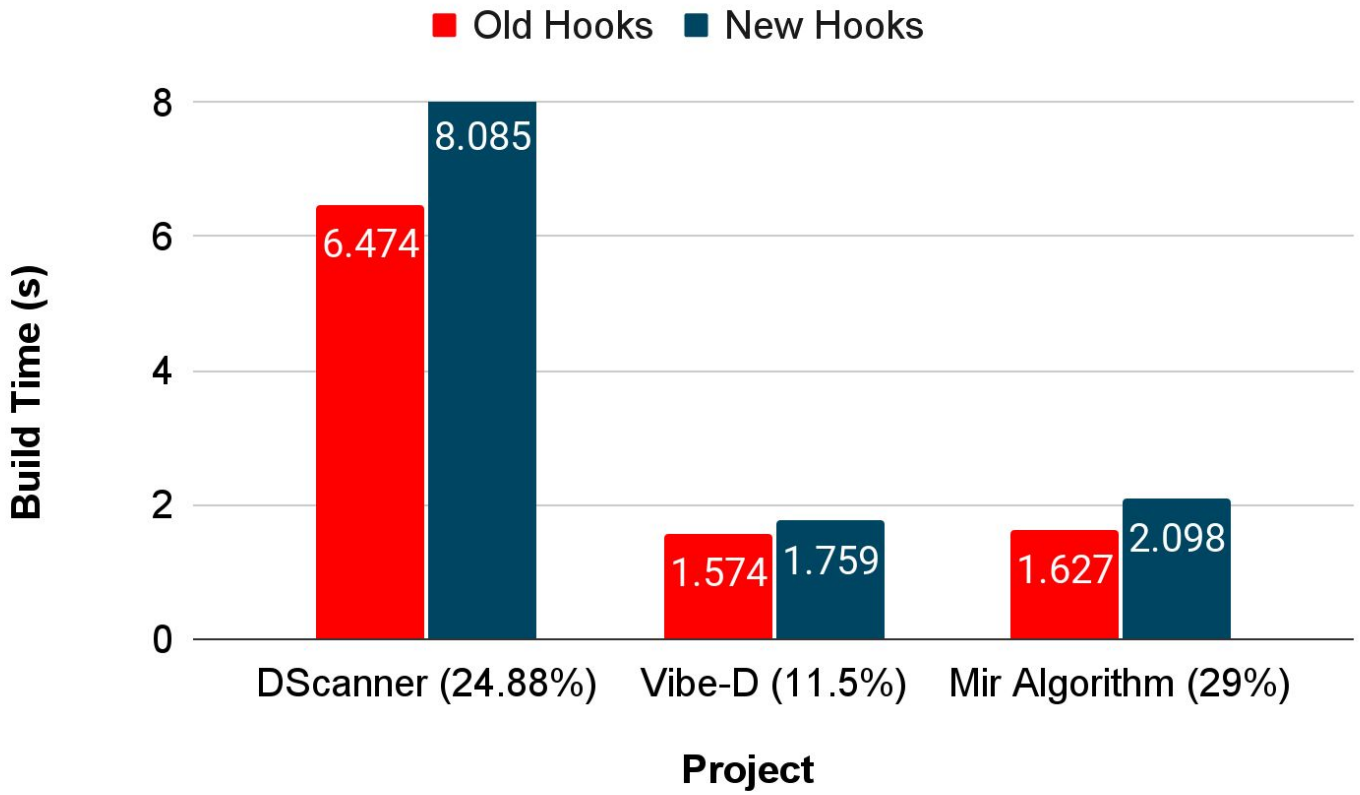
_d_arrayliteralTX
_d_assocarrayliteralTX
_d_arraysetcapacity
_d_arraysetlengthiT
_d_arraysetlengthT
_d_arrayshrinkfit
_d_interface_cast
_d_isbaseof
_d_isbaseof2

Performance Increase - All Array Hooks



https://github.com/teodutu/druntime-hooks-benchmarks

Compilation Times



Takeaways

- SAoC 2021: Exploration
- SAoC 2022: Fine Tuning
- SAoC 2023: Revolution