

Large-Scale C/C++ Code Restructuring with Coccinelle or, C/C++ refactorings on steroids

Michele MARTONE¹, Julia LAWALL²

¹ Leibniz Supercomputing Centre, Garching near Munich, Germany

² Inria Paris, France

Cross-Platform Development with C/C++ track
at deRSE24

Conference for Research Software Engineering in Germany
Würzburg University, Germany, March 06, 2024

Suppose you have a

- ▶ large C simulation code base ($> 100\text{kLoC}$)
- ▶ developed for many years, by many people

Lots of loop accesses

E.g.

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

What could be most obvious optimization (for CPUs)?

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

What could be most obvious optimization (for CPUs)?

Hint: `S[i]` are structures

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

Change AoS into Structure-of-Arrays (SoA)

```
for(i = 0; i < N; ++i)
{
    S[i].Metals = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = S[i].Metals * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

⇒

```
for(i = 0; i < N; ++i)
{
    SoA.Metals[i] = S[i].Abundance * factor + S[i].DeltaMetals;
    S[i].Abundance = SoA.Metals[i] * P[i].InvMass;
    S[i].DeltaMetals = 0;
}
```

Ok but how?

Ok but how?

- ▶ regexps (sed, awk, perl, python...)?

Ok but how?

▶ regexps (sed, awk, perl, python...)?

▶ a *source-to-source translator*?

LLVM?

ROSE?

Ok but how?

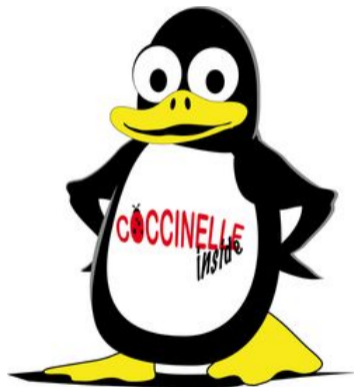
▶ regexps (sed, awk, perl, python...)?

▶ a *source-to-source translator*?

LLVM?

ROSE?

COCCINELLE!!



Originally, *to identify* and *to eliminate* bugs



AoS to SoA in a Nutshell

```
1 @@
2 identifier q =~ "prsr";
3 identifier s = {P};
4 identifier i = {i,k};
5 fresh identifier n = s ## "_" ## q;
6 @@
7 - s[i].q
8 + n[i]
```

basicaos2soa1.cocci

```
0 @@ -1,9 +1,9 @@
1 struct p_t {
2     double prsr, vel;
3 };
4
5 p_t P[3];
6 int main() {
7     int i,j,k;
8 -     P[i].prsr++;
9 +     P_prsr[i]++;
10 }
```

basicaos2soa1.diff

Managing complexity

- ▶ GPU *language extensions* (CUDA, HIP, ...)
- ▶ modern C++ features (mdspan, SYCL, ...)
- ▶ advanced `#pragma` manipulation: OpenMP, ...
- ▶ enforcing C++ coding guidelines

New: CUDA GPUs extensions

```
1 #spatch --smpl-spacing
2 @@
3 attribute name __device__;
4 @@
5 + __device__
6   int f(void) {
7 +   const int gthid =
8 +     blockIdx.x * blockDim.x
9 +       + threadIdx.x;
10     ...
11 }
12
13 @@
14 identifier d,b;
15 @@
16   int d;
17   int b;
18 - f();
19 + f<<<d,(d+b-1)/b>>>();
```

```
0 @@ -1,10 +1,13 @@
1 -int f(void)
2 +__device__ int f(void)
3 {
4 +   const int gthid =
5 +     blockIdx.x * blockDim.x
6 +       + threadIdx.x;
7   { /* loops */ }
8 }
9 int main(void)
10 {
11   int ds;
12   int bs;
13 - f();
14 + f<<<ds,(ds+bs-1)/bs>>>();
15 }
```

cuda.diff

New: C++23 multi-index operators

```
1 # spatch --c++=23
2 @tomultiindex@
3 symbol a,i,j;
4 expression k;
5 @@
6 - a[i][j][k]
7 + a[i, j, k]
8
9 @@
10 symbol b;
11 @@
12 - b[...]
13 + b[0]
```

```
0 @@ -1,8 +1,8 @@
1 int main()
2 {
3     int a[1][1][1];
4     int b[1][1][1];
5     int i=0,j=0,k=0;
6 - a[i][j][k]++;
7 - b[i][j][k]++;
8 + a[i, j, k]++;
9 + b[0][j][k]++;
10 }
```

mdspan1.diff

New: decluttering with OpenMP 5.1 pragmas

```
1 @@
2 identifier c, f, l;
3 expression b, k;
4 type T;
5 @@
6
7 + const T j = k;
8 + #pragma omp unroll partial (j)
9   for (T c=0; c
10 -           + k - 1
11             < l ; ... )
12 {
13 -     f(b+0);
14 -     f(b+1);
15 -     f(b+2);
16 -     f(b+3);
17 +     f(b);
18 }
```

deunroll1.cocci

```
0 @@ -1,12 +1,11 @@
1 int f(int i){}
2 int main()
3 {
4     const int n = 13;
5 -     for (int i=0;i+4-1<n;i++)
6 +     const int j = 4;
7 +#pragma omp unroll partial (j)
8 +     for (int i=0;i<n;i++)
9     {
10 -         f(i+0);
11 -         f(i+1);
12 -         f(i+2);
13 -         f(i+3);
14 +         f(i);
15     }
16 }
```

deunroll1.diff

C++ Core Guideline F.16 (Bjarne Stroustrup, Herb Sutter)

```
1 #spatch --c++
2 @r1@
3 type T;
4 identifier f;
5 parameter list pl;
6 @@
7 T f(pl) { ... }
8
9 @r2@
10 typedef A, B;
11 type heavy_type={A,B};
12 type r1.T;
13 identifier r1.f;
14 symbol i;
15 @@
16 +// Note: heavy copy!
17   T f (
18     ..., heavy_type i,
19     ...
20   ) { ... }
```

```
0 @@ -1,13 +1,16 @@
1 #include <array>
2 struct A { std::array<int,999> a; }; // heavy
3 struct B { std::array<int,999> a; }; // heavy
4 struct C { std::array<int, 16> a; }; // light
5 void if1(int i) {}
6 void if2(int &arg) {}
7 void if3(const int &arg) {}
8 +// Note: heavy copy!
9 void af1(A i) {}
10 +// Note: heavy copy!
11 void bf1(B i) {}
12 +// Note: heavy copy!
13 void bf2(const B i) {}
14 void bf3(const B & i) {}
15 void cf1(C i) {}
16 int main() { }
```

cppcg_f16.diff

No Raw Loops (Sean Parent)

```
1 #spatch --c++=17
2 @@ @@
3 #include <iostream>
4 + #include <algorithm>
5 + #include <functional>
6
7 @@
8 type T;
9 constant k;
10 identifier elem,result,arrid;
11 @@
12 - bool result = false;
13 ...
14 - for ( T &elem : arrid )
15 - if ( !( elem == k || k == elem ) )
16 - {
17 - ...
18 - result = true;
19 - break;
20 - }
21 + const bool result =
22 + (find(begin(arrid),end(arrid),k) !=
23 + end(arrid));
```

norawloops.cocci

```
0 @@ -1,20 +1,15 @@
1 #include <vector>
2 #include <iostream>
3 + #include <algorithm>
4 + #include <functional>
5 int main()
6 {
7     using namespace std;
8     vector v = {1,2,3};
9 - bool has_zero = false;
10
11     v[2] = 0;
12
13 - for ( int &a : v )
14 - if ( 0 == a )
15 - {
16 -     cout << "doing things\n";
17 -
18 -     has_zero = true;
19 -     break;
20 - }
21 + const bool has_zero =
22 + (find(begin(v), end(v), 0) != v.end());
23     cout << has_zero << endl;
24 }
```

norawloops.diff

Currently working on

- ▶ covering C++ features
- ▶ further use cases

Acknowledgements

Work made possible by


▶ SiVeGCS Project



▶ BayFrance travel grant



 **Tutorial this afternoon 15:50–17:10, Room SE2**

 Coccinelle's GitHub page

<https://github.com/coccinelle/coccinelle>