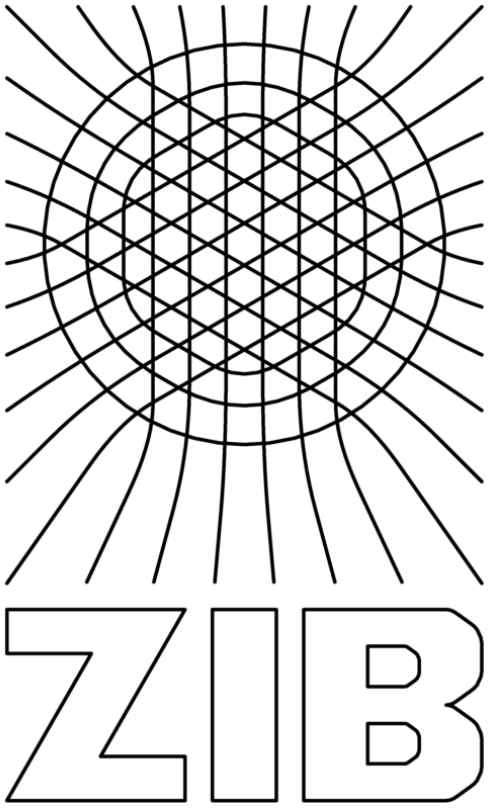
An Automated Approach for Estimating the Memory Footprint of Non-Linear Data Objects



Sebastian Dreßler
Thomas Steinke
Zuse Institute Berlin

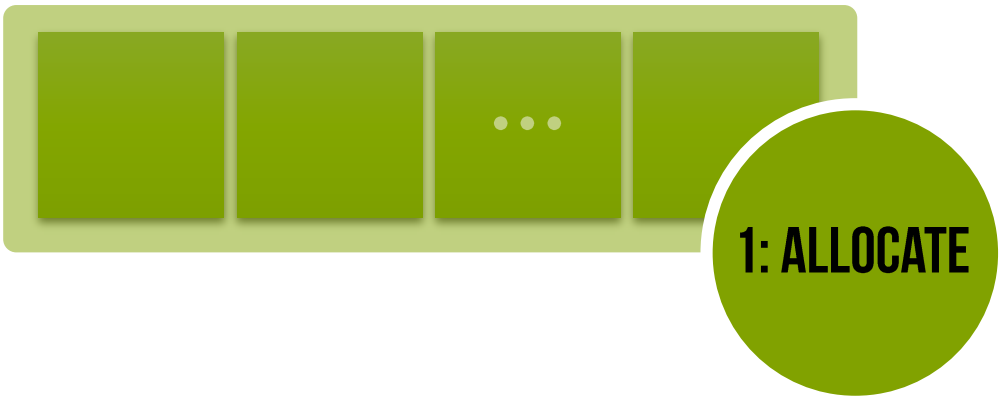
HETEROGENEOUS SYSTEMS: MANUAL SERIALIZATION PRIOR TRANSFERS OFTEN NEEDED

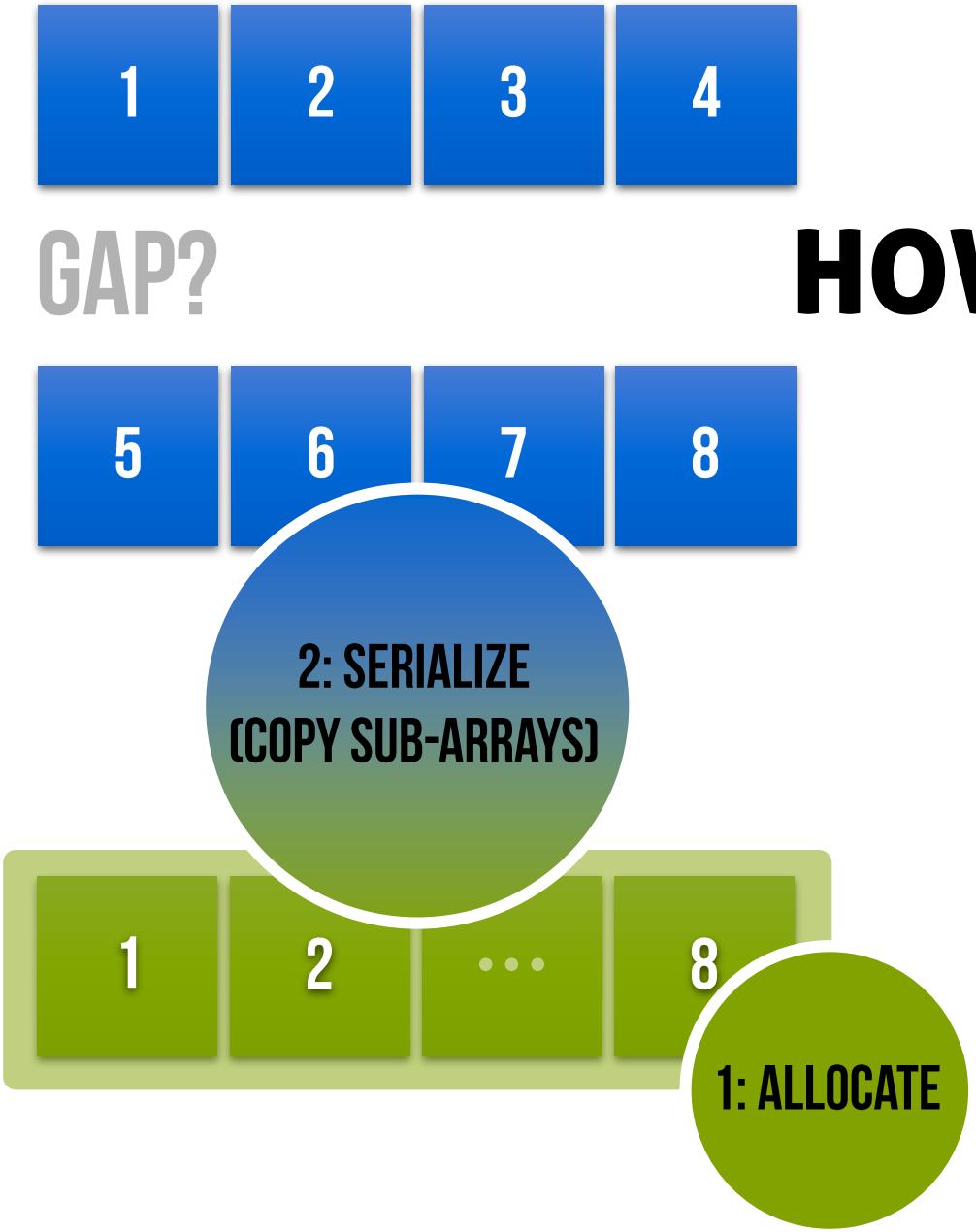
GAP?

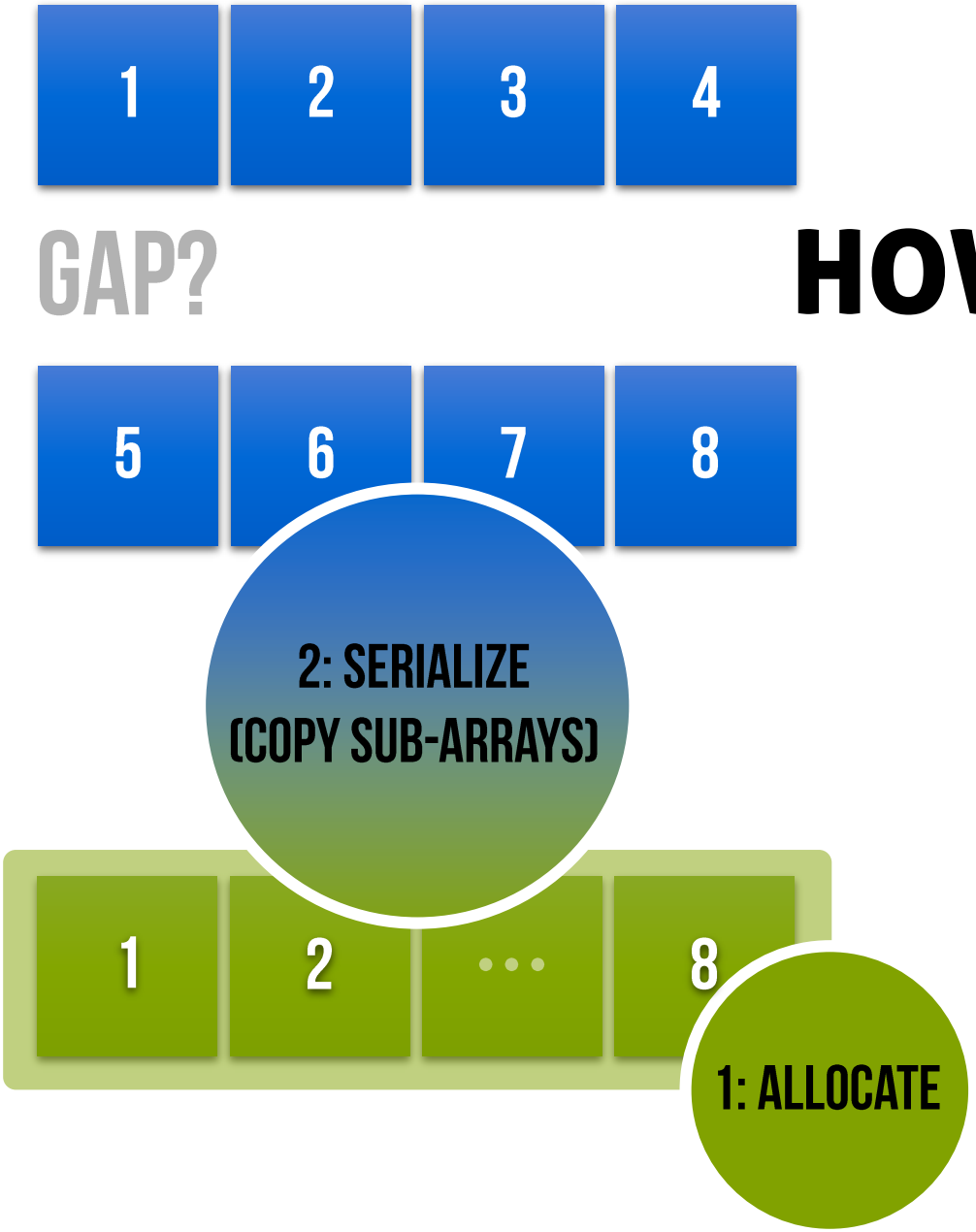
HOW TO COPY?

GAP?

HOW TO COPY?




1
2
3
4
GAP?
HOW TO COPY?
5
6
7
8
2: SERIALIZE
(COPY SUB-ARRAYS)
1
2
...
8
1: ALLOCATE



HOW TO COPY?

MANUAL TASKS

„BUT THERE ARE ALTERNATIVES LIKE #PRAGMA-DIRECTIVES!"

YES, BUT

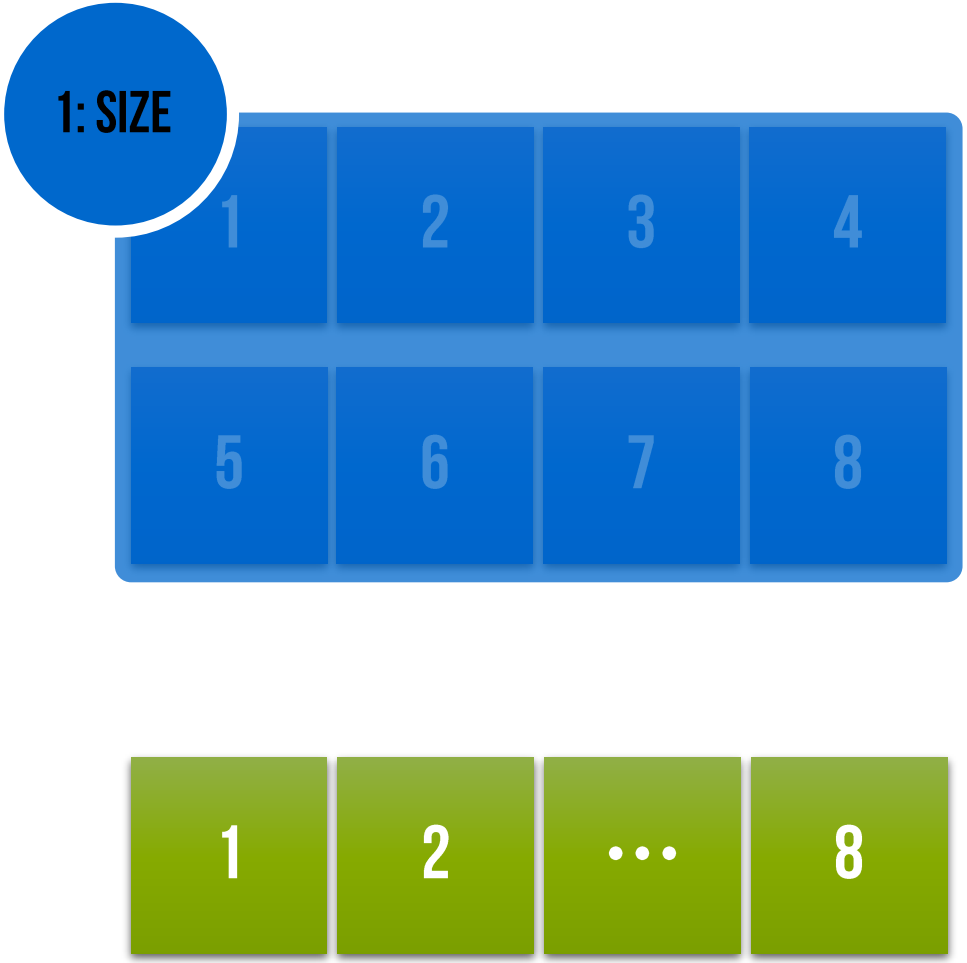
„Still manual work“

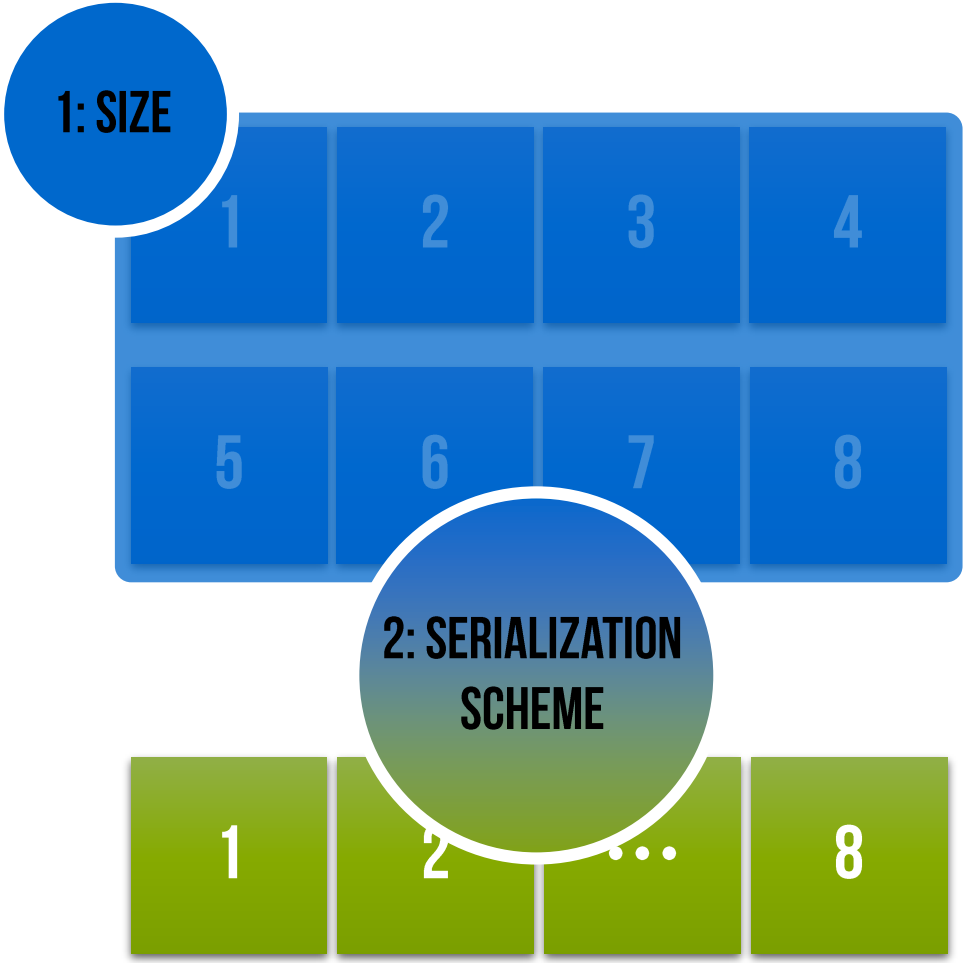
How „dynamic“ they are?

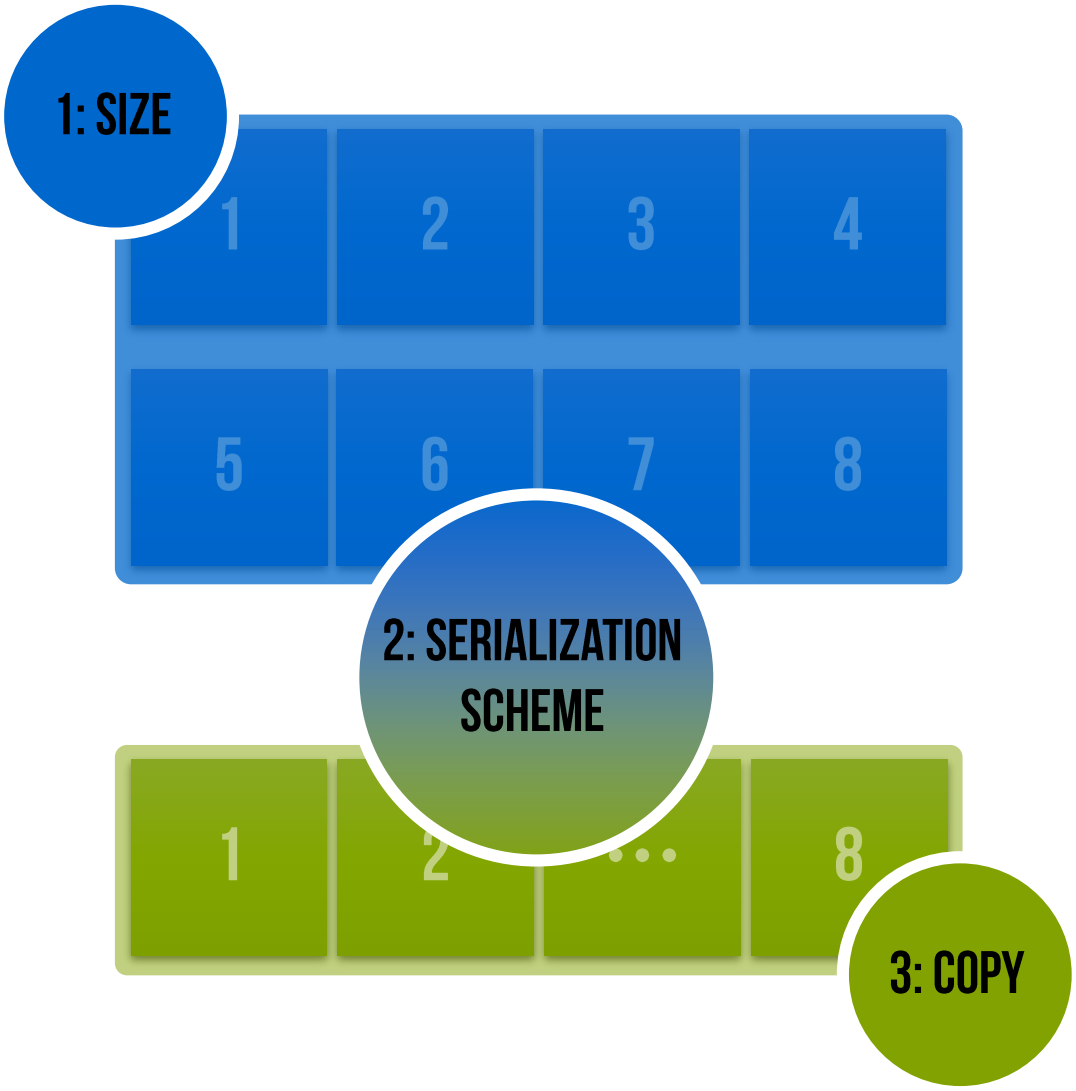
Support for non-linear objects?

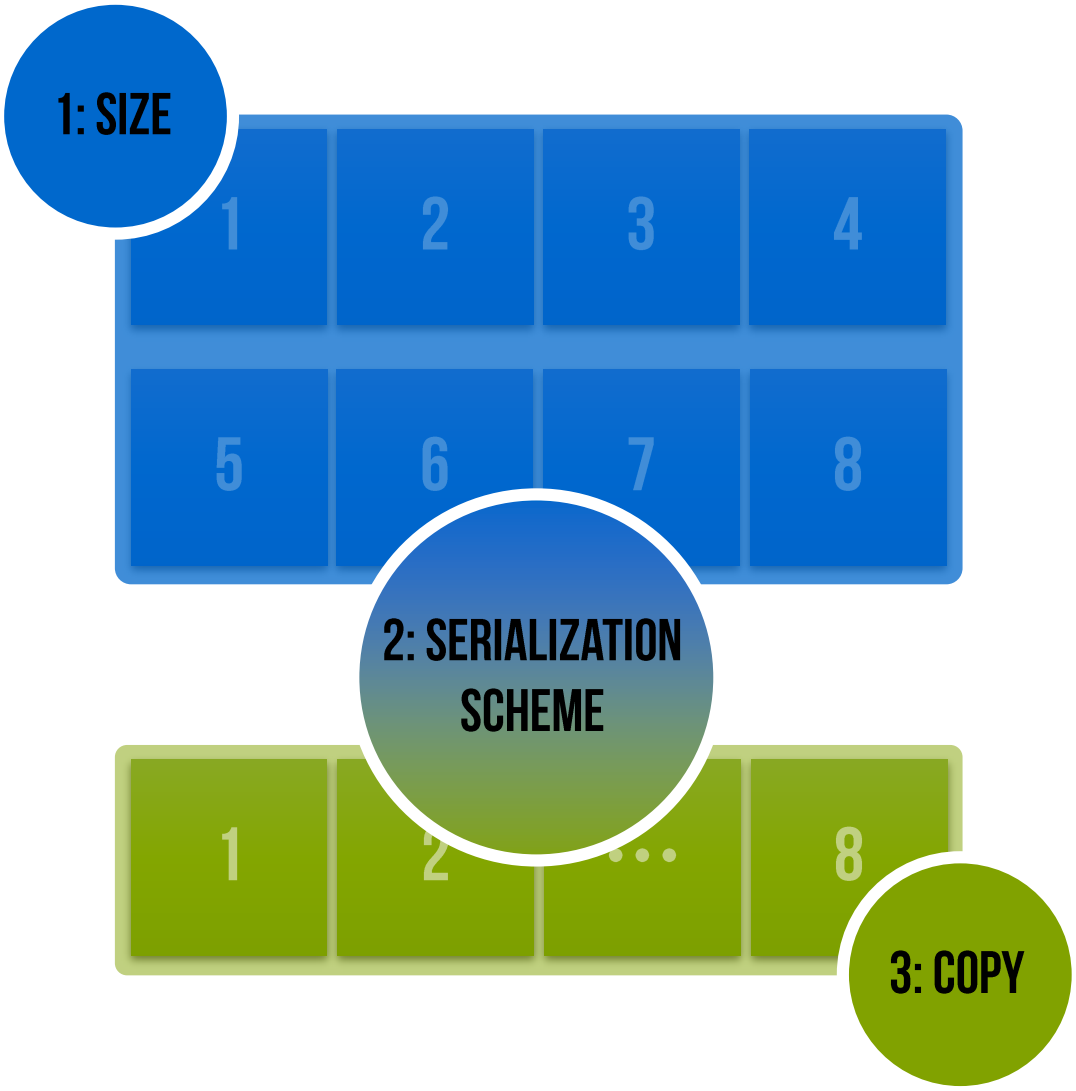
AUTOMATED TRANSFERS, A SOLUTION PROPOSAL (IN THREE STEPS)


1
2
3
4
5
6
7
8
1
2
…
8


1: SIZE
1
2
3
4
5
6
7
8
1
2
…
8


1: SIZE
1
2
3
4
5
6
7
8
2: SERIALIZATION SCHEME
1
2
...
8


1: SIZE
1
2
3
4
5
6
7
8
2: SERIALIZATION SCHEME
1
2
...
8
3: COPY



METHODS ARE AUTOMATED

OUR PAPER COVERS STEP 1: MEMORY FOOTPRINT (+DATA DIRECTION)

APPROACH: LLVM & GRAPHS

Language independent

Extensible

Information traversal eased

Supports non-linear objects

WORKFLOW

1. Generate DD Graph
2. Extract data direction
3. Generate TD Graph
4. Inject size functions into code

EXAMPLE CODE

```
void F(std::vector<std::vector<int>> &X, int N) {
  for (int i = 0; i < N; i++) {
    X = TF(X);
  }
}
```

DATA DEPENDENCY GRAPH

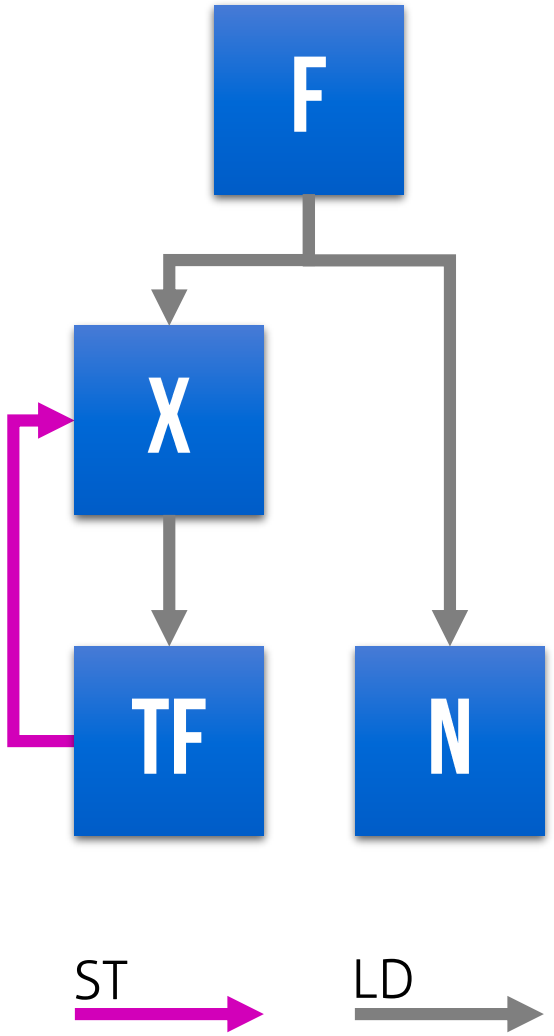
Directed

Object Loads / Stores

Explore function calls

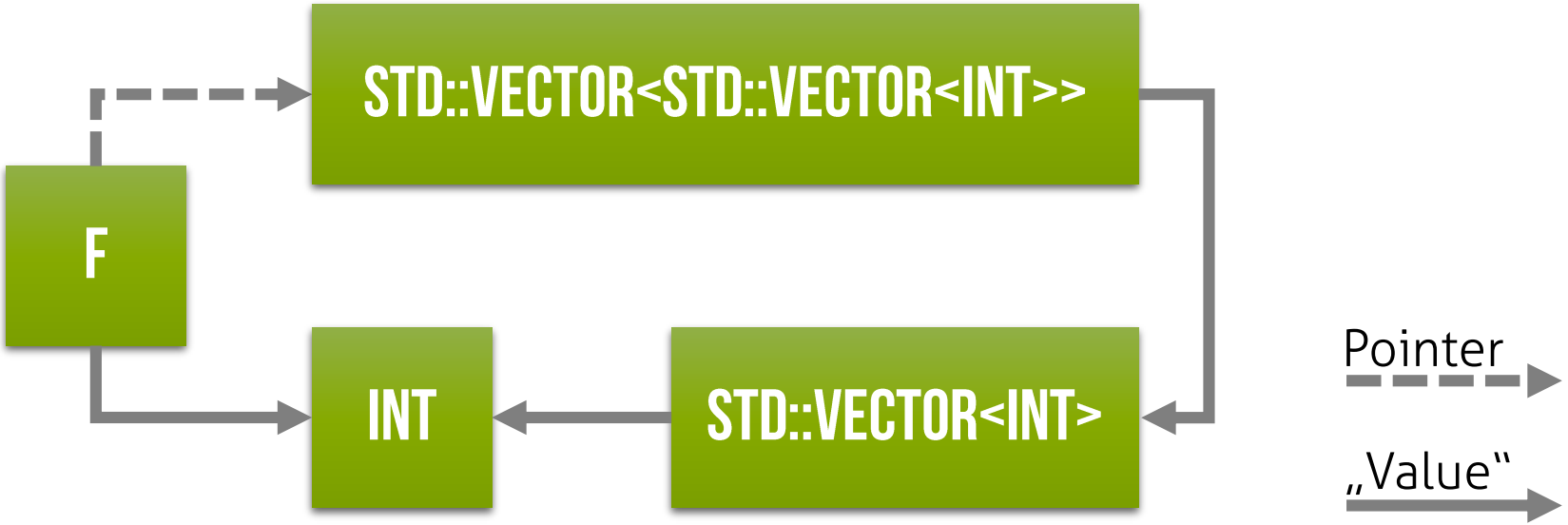
N : LD Only » IN

X : LD + ST » IN + OUT



TYPE DEPENDENCY GRAPH

Directed
Object type breakdown
Recursion with cycles



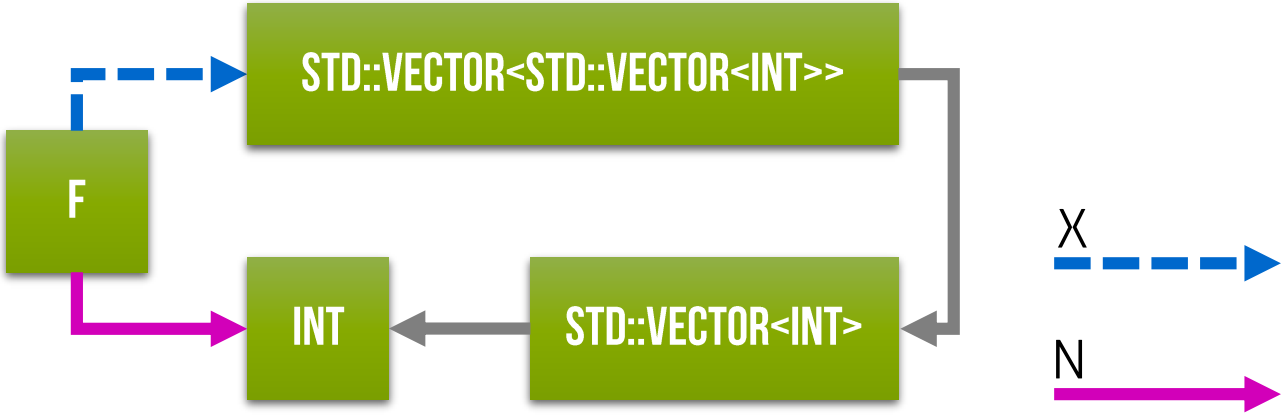
SIZE FUNCTIONS...

Directly written in LLVM IR

Statically injected

Consider structural type information

Are dynamic due to run-time execution




```
sizeof(X) of type std::vector<std::vector<int>> is
  s = 0
  foreach x in X:
    s += count(x) * sizeof(int)

  return s
```

```
sizeof(N) of type int is
  return sizeof(int)
```

SIZE FUNCTIONS AND TYPES

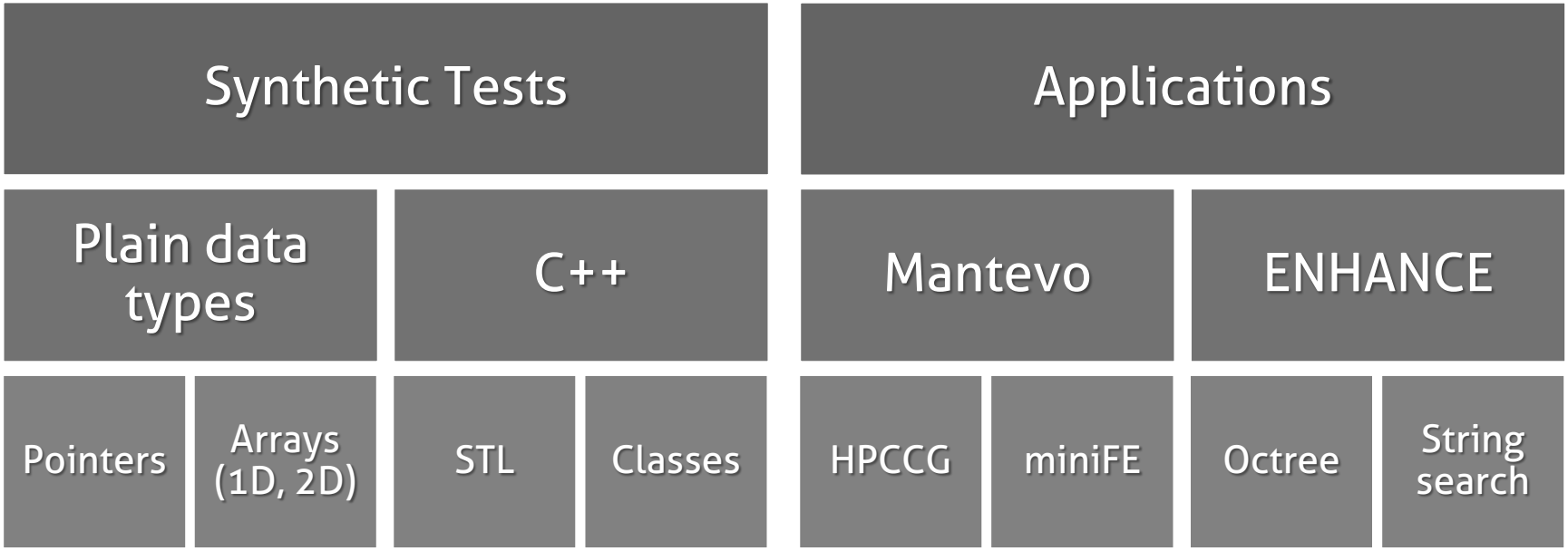
```
class A contains { int X, Y, Z; string name; }
```

```
sizeof(Y) of type std::vector<class A> is
  s = 0
  foreach y in Y:
    s += sizeof(y)

  return s
```

```
sizeof(a) of type class A is
  return 3 * sizeof(int) + sizeof(a.name)
```

```
sizeof(s) of type string is
  return count(s) * sizeof(char)
```

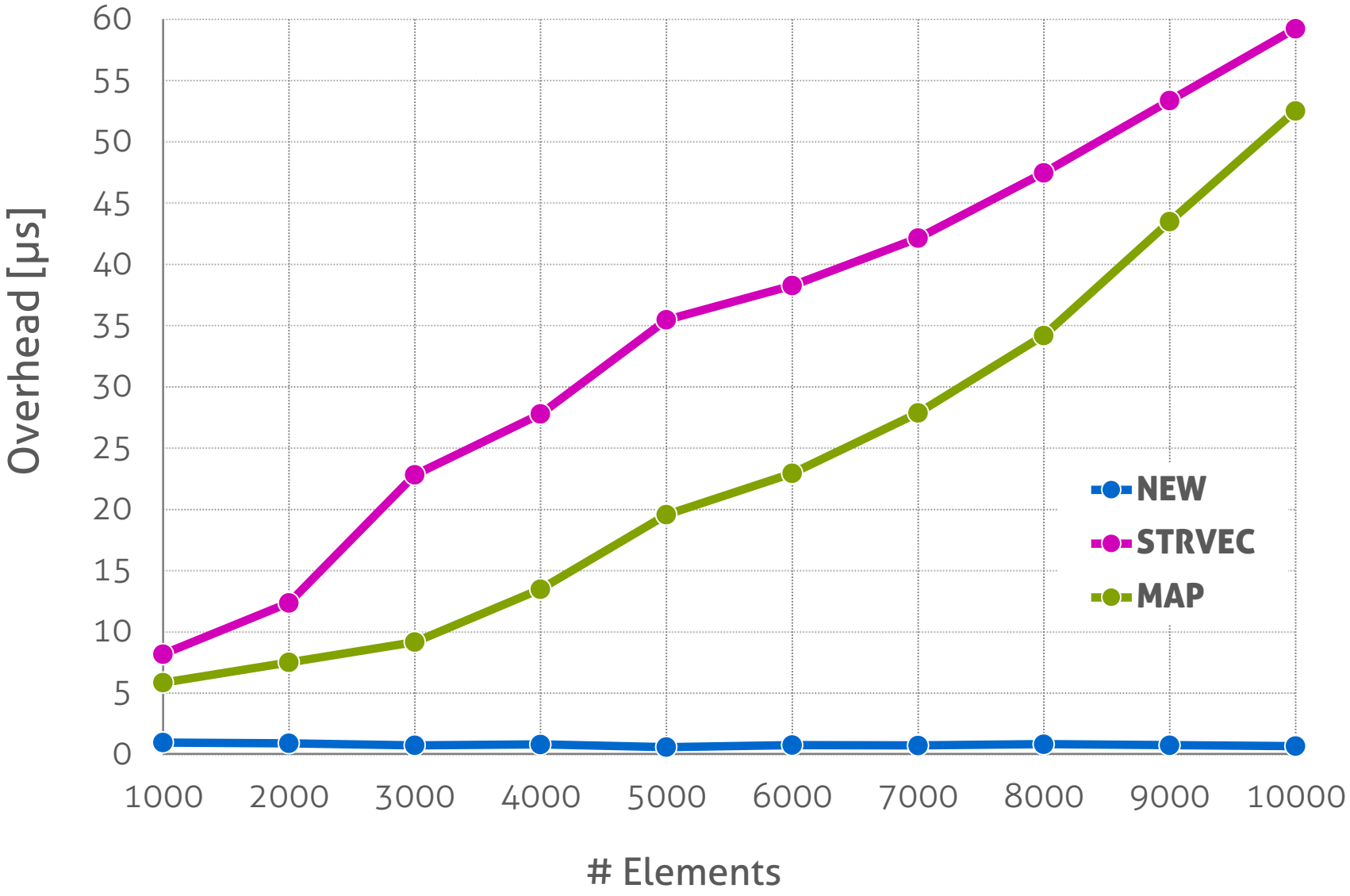


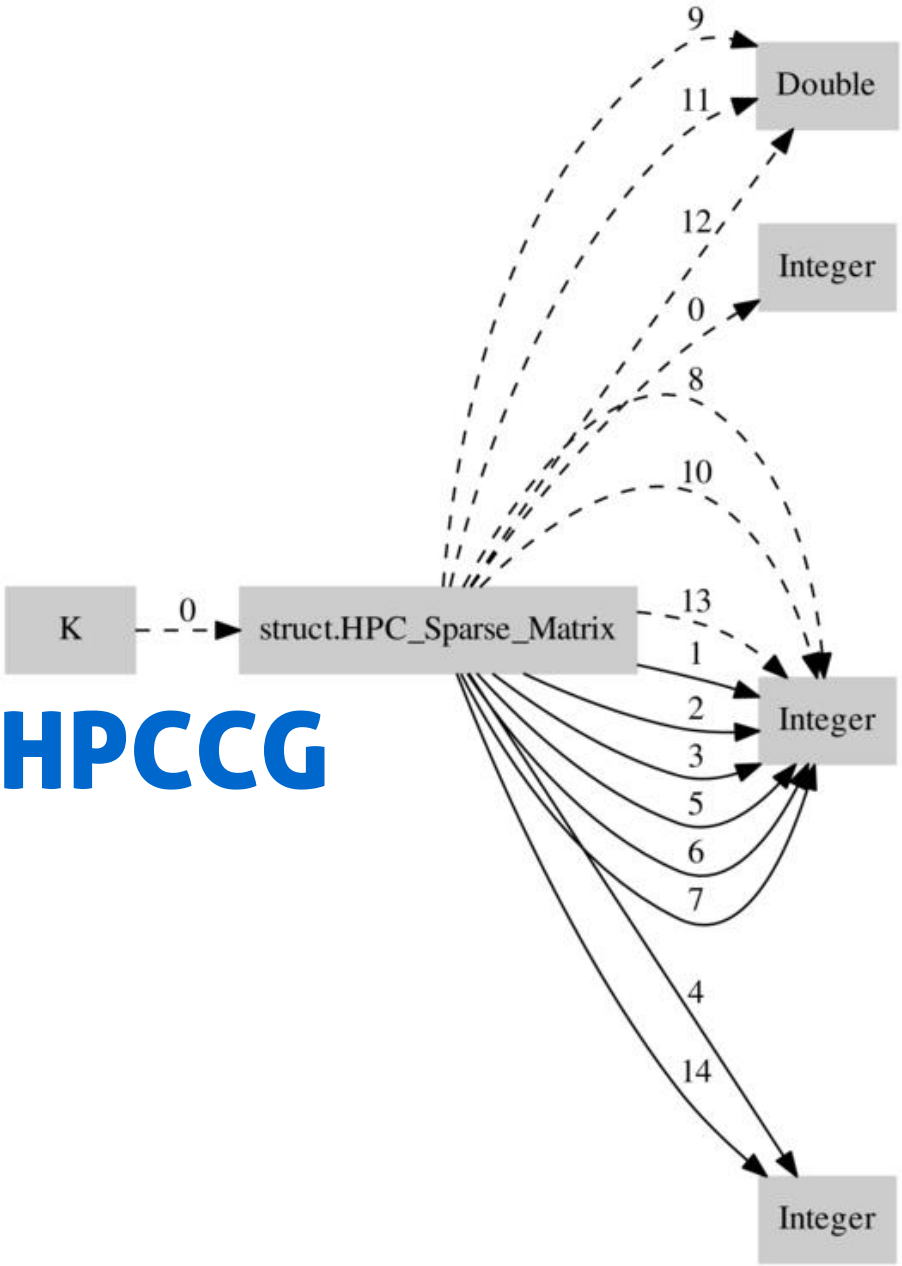
EVALUATION / LIMITATIONS

Overlapping pointers

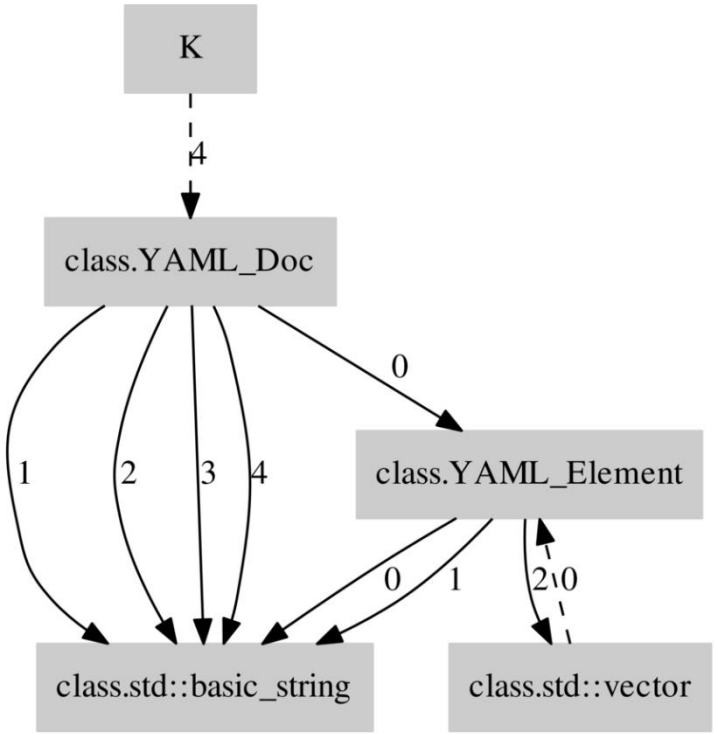
Function pointers

Re-allocation


Overhead [µs]
60
55
50
45
40
35
30
25
20
15
10
5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Elements
NEW
STRVEC
MAP

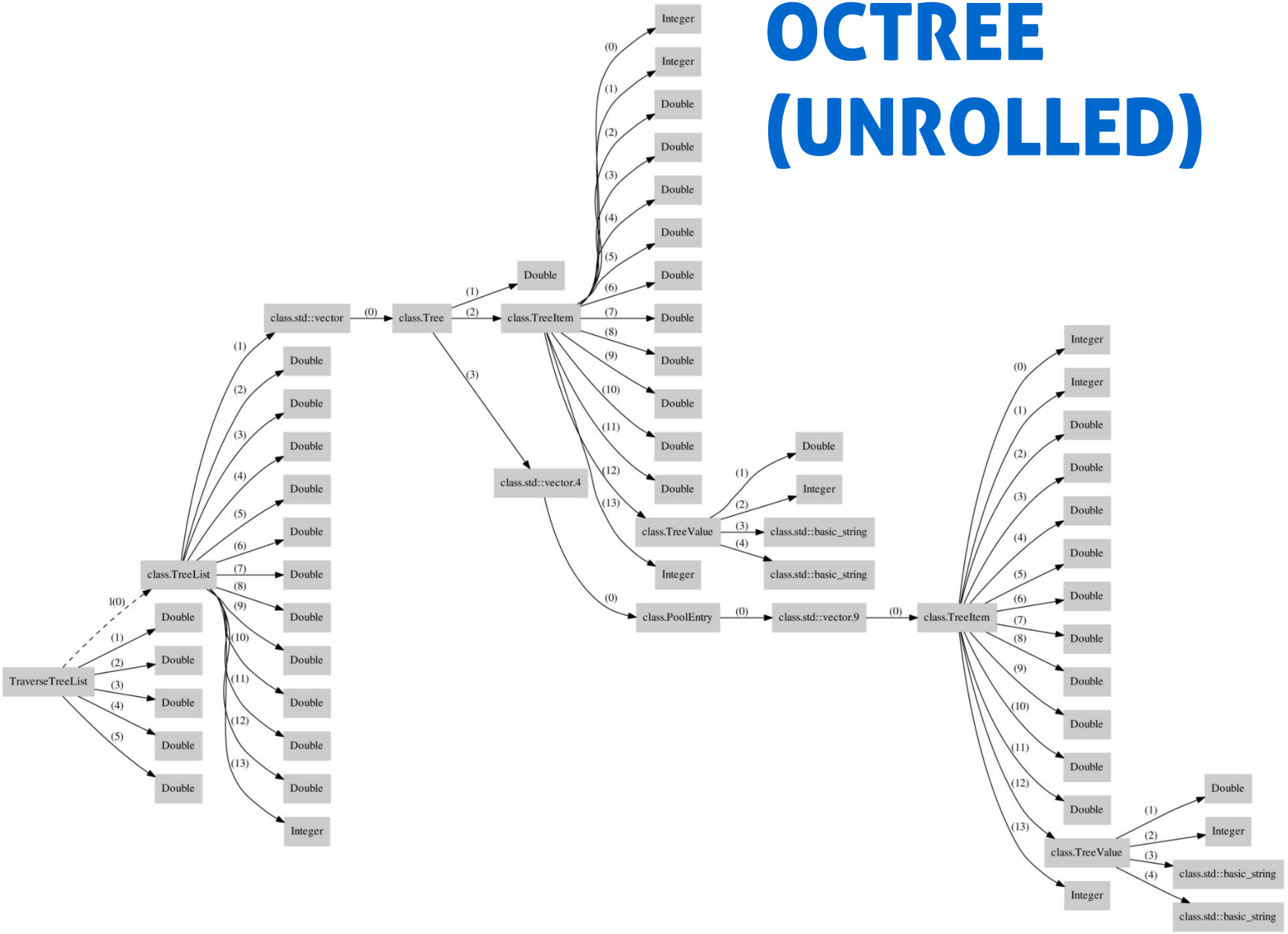


HPCCG



MINIFE

OCTREE (UNROLLED)



CONLUSION

Automatic memory footprint / data direction

LLVM & Graphs

Support for non-linear objects

OUTLOOK

Improve type compatibility

Step 2: analyze host / device layout

Step 3: copy w.r.t. data layout

SPONSORED BY THE

Federal Ministry
of Education
and Research

Grant No. 01|H11004G

THANK YOU.