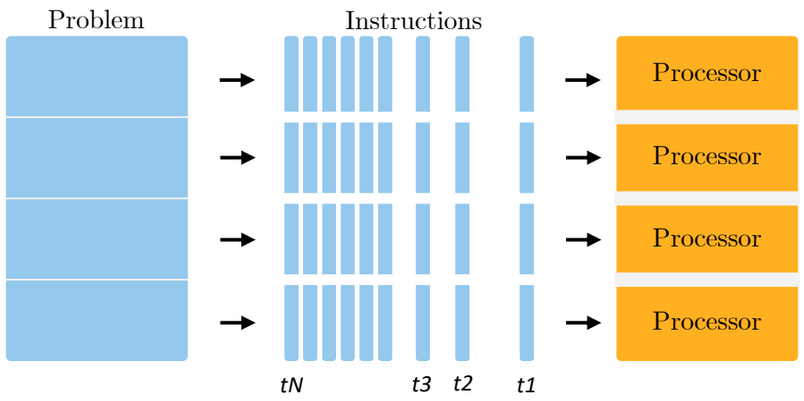
Can We Run In Parallel? Automating Loop Parallelization For TornadoVM

Shreyansh Kulshreshtha†, Rishi Sharma† and Manas Thakur†

Indian Institute of Technology Mandi

Parallelization



- Speedup execution utilizing Parallelism.
- Parallelize hot portions – *Loops*.
- Must not contain loop carried dependence.

Limitations of TornadoVM

```
1 public void scale(int[] ar) {
2   int n = ar.length;
3   int alpha = 2;
4   for (int i = 0; i < n; i++) {
5     ar[i] = ar[i]*alpha;
6   }
7 }
```

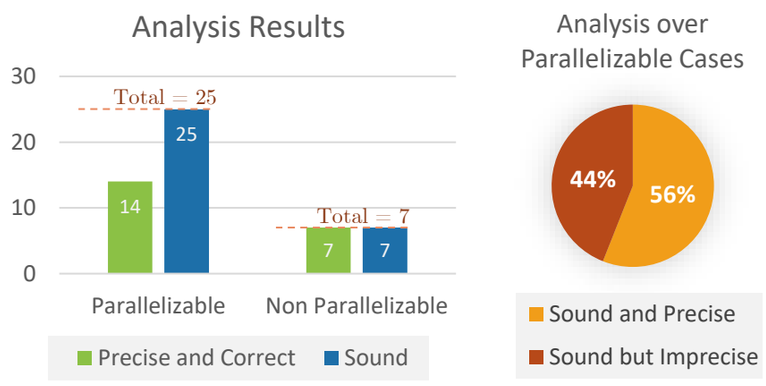
- Does not automatically insert annotation at potential locations.
- Users have to identify parallelizable loops.

```
1 public void swap(int[] ar) {
2   int n = ar.length;
3   for (@Parallel int i = 0; i < n; i++) {
4     int temp = ar[i];
5     ar[i] = ar[i-1];
6     ar[i-1] = temp;
7   }
8 }
```

- Does not verify if the annotated loop is parallelizable.
- Wrongly annotated loops can lead to unsound results.

Results

Evaluating Precision and Soundness of AutoTornado over Test Cases



Test Program	Running Time (seconds)		Analysis Time (seconds)
	Before AutoTornado	After AutoTornado	
Saxpy	22.205	2.224	2
HilbertMatrix	9.981	3.842	2
MatrixTranspose	96.334	7.33	1
Convolution2D	27.982	5.152	1
VectorAddInt	10.322	1.742	1

Table - Running times of different programs in serial and parallel execution.

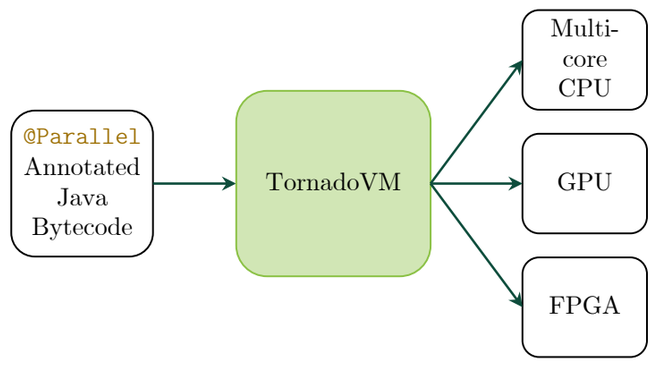
- Tests include examples provided by TornadoVM as well as self-written cases.

TornadoVM

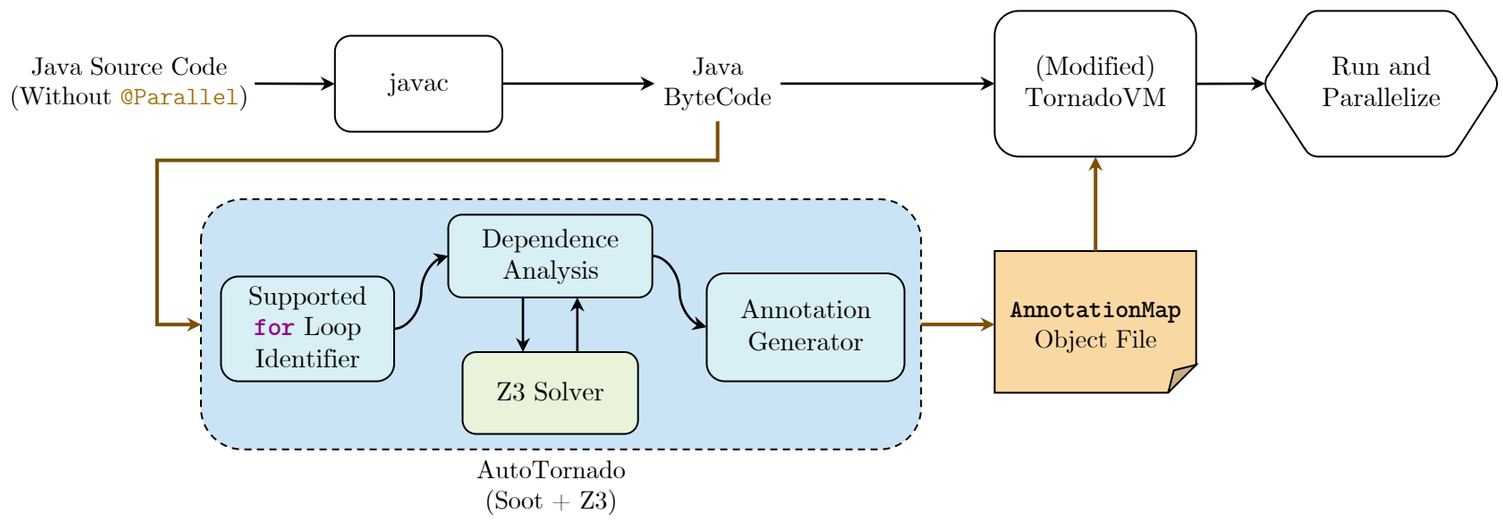
```
1 for (@Parallel int i = 0; i < n; ++i) {
2   // Do something
3 }
```

An annotated **for** loop

- TornadoVM[1]: A Java accelerator plugin for Java Virtual Machines.
- Parallelizes manually annotated loops.



Our Solution : AutoTornado



- Program analysis written in Soot[2].
- Supported Loop Identifier -
  - Identify bounds, update statement and iteration variable.
  - Reject loops with multiple update statements or exits.
  - Not trivial because loops in *Jimple* (Soot's IR) composed of **if**s and **goto**s.
- Dependence Analysis -
  - Identify scalar variables not local to the loop (**nonLocalVars**).
  - Object references treated as **nonLocalVars**.
  - Loops having function calls or writes to **nonLocalVars** rejected.
  - Array references handled separately -
    - Array elements written to in one iteration should not be read from or written to in any other iteration.
    - Z3 Solver[3]: Identify array dependence by solving the *Satisfiability Problem*.
    - Encode relevant portions of the program into logic.

```
1 public void foo(int ar[]) {
2   for(int i=0; i<10000; i++) {
3     int k1 = f1(i);
4     int k2 = f2(i, k1);
5     int k3 = f3(i, k2);
6     ar[k3] = k2;
7   }
8 }
```

$(k3^u == f3(i^u,k2^u)) \wedge (k2^u == f2(i^u,k1^u)) \wedge (k1^u == f1(i^u)) \wedge$
$(k3^v == f3(i^v,k2^v)) \wedge (k2^v == f2(i^v,k1^v)) \wedge (k1^v == f1(i^v)) \wedge$
$(i^u \geq 0) \wedge (i^u < 10000) \wedge (i^v \geq 0) \wedge (i^v < 10000) \wedge$
$(i^u \neq i^v) \wedge (k3^u == k3^v)$ u and v are different iterations $(u \neq v)$

Problem Statement – "*Is there a satisfying assignment for variables under the given constraints, such that the index can be the same for two different iterations?*"

- Annotation Generator -
  - Extract *start_pc*, *slot* and *length* attributes of iteration variable from the class file.
  - Write to **AnnotationMap** object file.

Conclusion & Future Work

- Conclusion:
  - AutoTornado identifies loop-carried dependences and marks appropriate loops as parallelizable.
  - Most of the loops that are marked not parallelizable conservatively come from the scalar analysis.
- Future work:
  - Plans to improve scalar analysis and handle function calls inside loops.

References

[1] Juan Fumero. 2020. TornadoVM: Accelerating Java with GPUs and FPGAs (2020). https://www.infoq.com/articles/tornadovm-java-gpu-fpga/

[2] Raja Vallée-Rai, Phong Co, Etienne Gagnon, Laurie Hendren, Patrick Lam, and Vijay Sundaresan. 1999. Soot - a Java Bytecode Optimization Framework. CASCON '99. IBM Press.

[3] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. TACAS'08/ETAPS'08. Springer-Verlag, Berlin, Heidelberg, 337–340.

30th International Symposium on Software Testing and Analysis (ISSTA 2021)

† Authors' address: {shreyanshkuls,rishi-sharma}@outlook.com manas@iitmandi.ac.in, Indian Institute of Technology Mandi, Himachal Pradesh, India