



Hydra LLVM: Instruction Selection with Threads

Min-Yih Hsu and Prof. Jenq-Kuen Lee,
Department of Computer Science, National Tsing-Hua University, Taiwan
{myhsu, jklee}@pplab.cs.nthu.edu.tw

Instruction Selection is SLOW!

By the rise of program complexity and some specific usages like JIT(Just-In-Time) compilation, compilation speed becomes more and more important in recent years. Instruction selection in LLVM, on the other hand, is the most time-consuming part among all the LLVM components, which can take nearly **50%** of total compilation time

Instruction Selector in Current LLVM

LLVM has a greedy based instruction selector, which is basically a bytecode interpreter

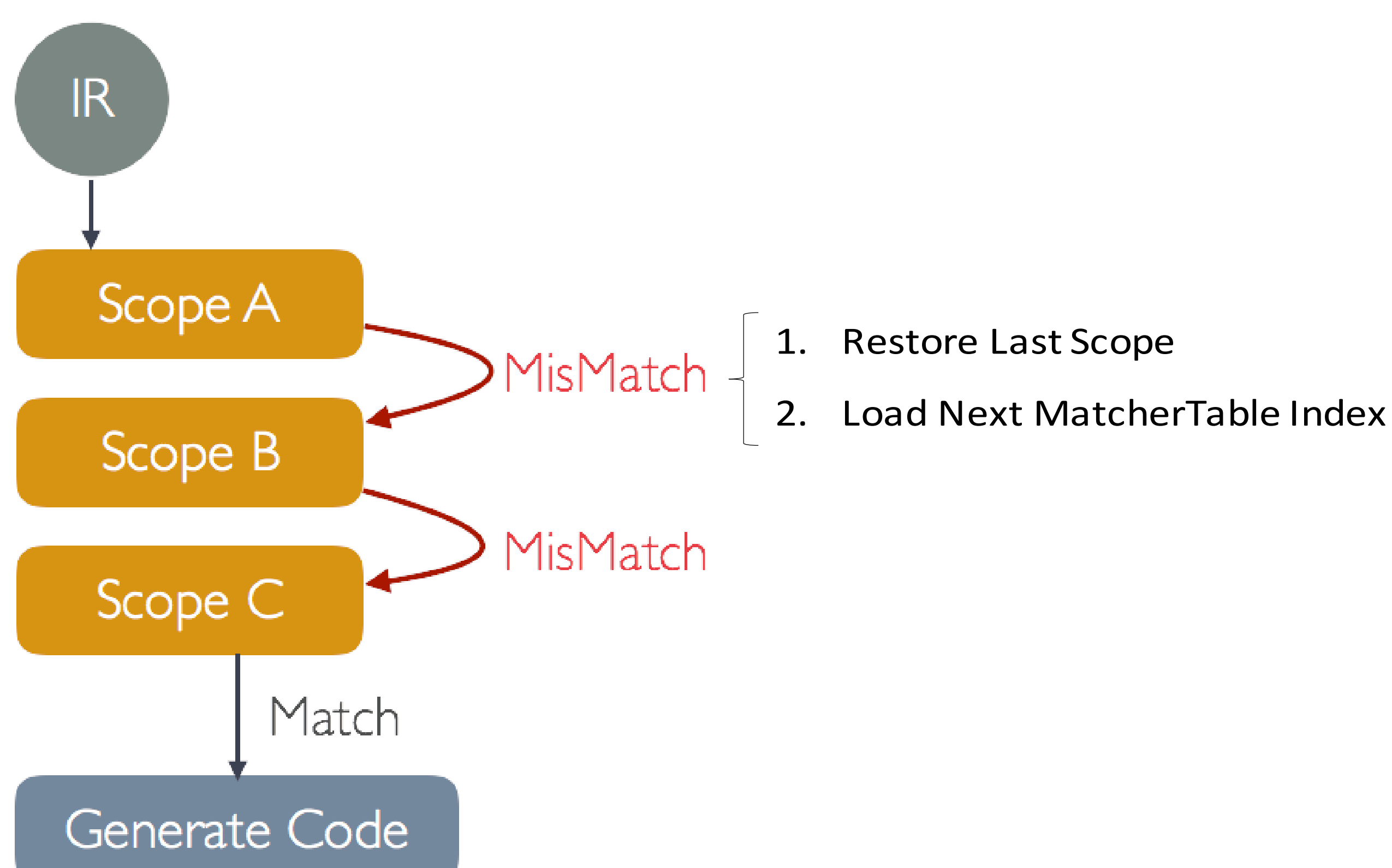
Scope: Save Context

```
12*/ OPC_Scope, 70, /*->84*/ // 17 children in Scope
14*/ OPC_CheckChild1Type, MVT::v4f32,
16*/ OPC_RecordChild2, // #2 = $dst
17*/ OPC_CheckPredicate, 0, // Predicate_unindexedstore
19*/ OPC_CheckPredicate, 1, // Predicate_store
21*/ OPC_CheckPredicate, 2, // Predicate_nontemporalstore
23*/ OPC_CheckPredicate, 3, // Predicate_alignednontemporalstore
25*/ OPC_Scope, 18, /*->45*/ // 3 children in Scope
27*/ OPC_CheckPatternPredicate, 0, // (Subtarget->hasAVX()) && (!Subtarget->hasVFX())
29*/ OPC_CheckComplexPat, /*CP*/0, /*#*/2, // selectAddr:$dst #3 #4 #5 #6 #7
32*/ OPC_EmitMergeInputChains1_0,
33*/ OPC_MorphNodeTo, TARGET_VAL(X86::VMOVNTPSnr), 0|OPFL_Chain|OPFL_MemRefs,
```

Checker Actions

Sub-Scope

In the interpreter, it uses checkers to perform pattern matching, and use scopes to provide a local context for a subset of checkers. Each scope would jump to next scope if a failure is raised within it.



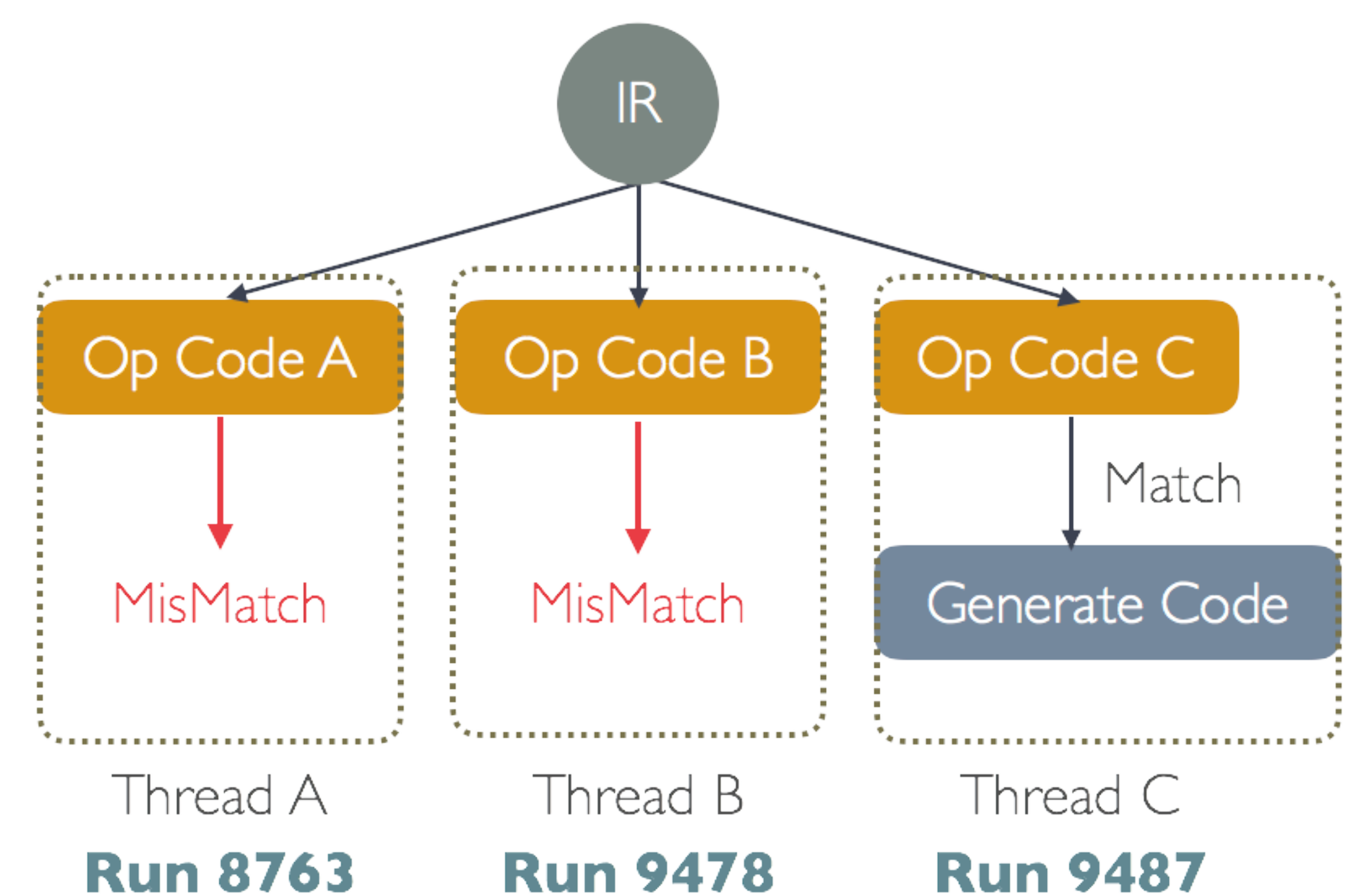
Parallelizing Selection Scopes with Threads

Each selection scope has the offset to the next scope, which is **statically known ahead-of-time**, thus provide us the basis of parallelizing the instruction selection process efficiently.

```
/*8763*/ OPC_Scope ... /*->9478*/
OPC_CheckXXX
...
/*9478*/ OPC_Scope ... /*->9487*/
...
/*9487*/ OPC_Scope ... /*->...*/
...
```

Offset of Next Scope

With the statically-known scope offset, we adopt a **speculation** based approach that execute every possible selection scopes **simultaneously** in threads



Experiment Results

