FAST PATH LOOP UNROLLING

Fast-Path Loop Unrolling of Non-Counted Loops to Enable Subsequent Compiler Optimizations

D. Leopoldseder, R. Schatz, L. Stadler, M. Rigger, T. Würthinger, H. Mössenböck

JOHANNES KEPLER UNIVERSITY LINZ

GraalVM https://www.graalvm.org/

Problem

```
for (int i = 0; i < upperBound; i++) {
  <loop body>
}
```
⇒ Iteration count known, easily optimizable

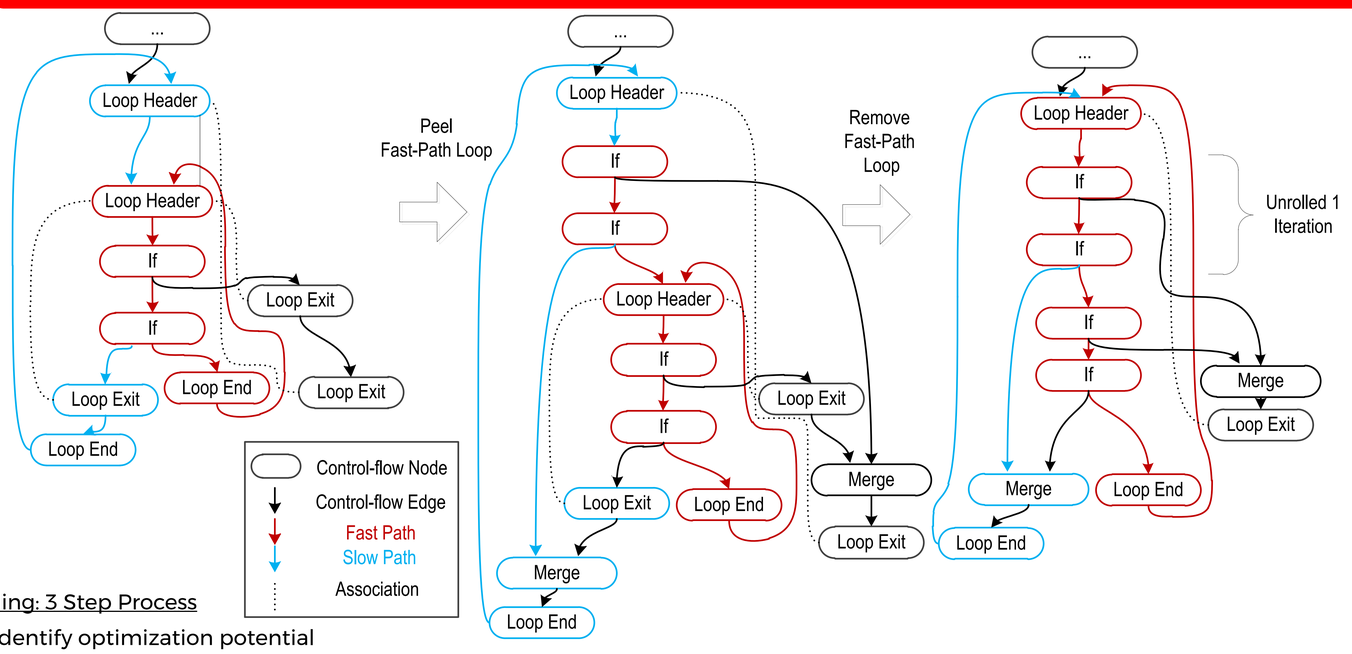
Counted loop

Non-Counted loop

```
while (condition) {
  <loop body>
}
```
⇒ Iteration count unknown, hard(er) to optimize

```
for (int i = 0; i < upperBound - (upperBound % 2); i += 2){
 <loop body>
 <loop body>          // Removed intermediate iteration checks
}
if (upperBound % 2 != 0) { <loop body> }   // Fixup iteration(s)
while (condition) {
 <loop body>
 if (condition) {     // Cannot remove iteration check
   <loop body>
   continue;
  }
 break;
}
```

Fast-Path Loop Unrolling



Unrolling: 3 Step Process

(1) Identify optimization potential

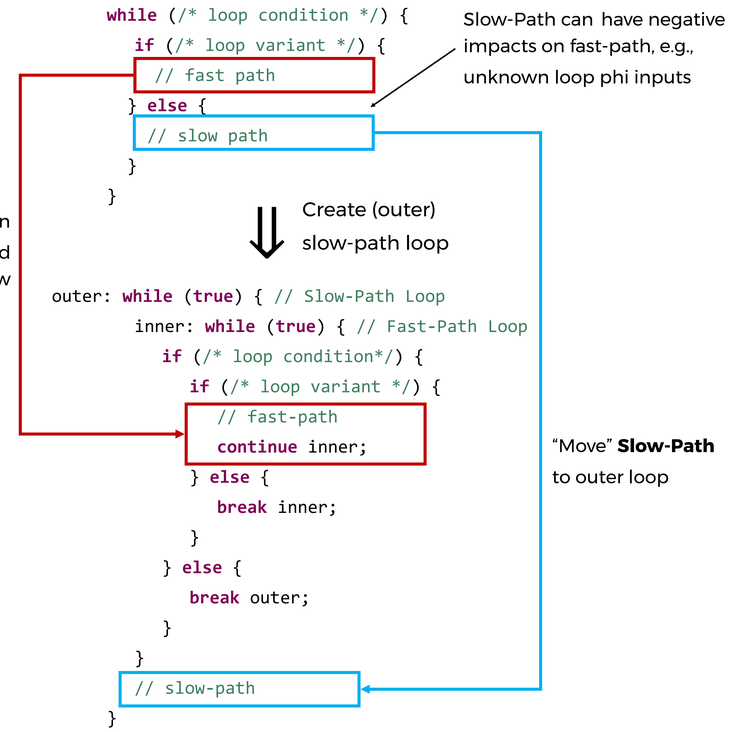
(2) Decide which loops are beneficial to unroll (code size vs performance trade-off)

(3) Perform 2 phase transformation

-> Peel fast-path loop

-> Remove fast-path loop

Fast-Path Loop Creation



Fast-Path can be better optimized now

Desirable to unroll

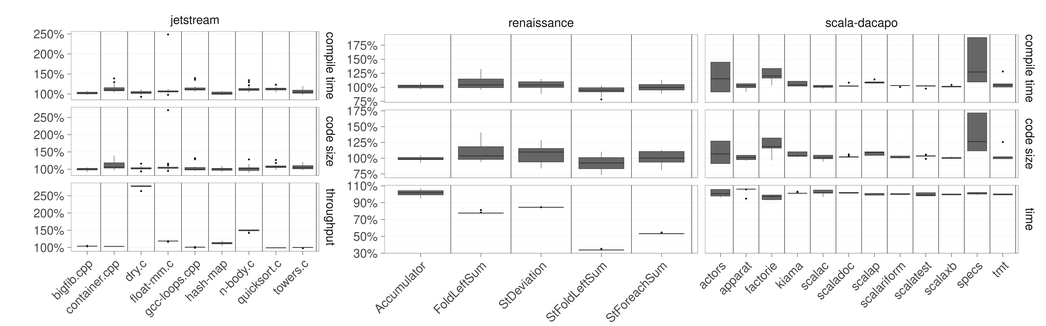
Enable other Optimizations

-> Safepoint poll reduction

-> Canonicalizations

-> Loop-carried dependencies

Performance



Performance relative to baseline configuration without unrolling