循环优化系列第二讲

先进编译实验室
Advanced Compiler

循环合并

先进编译实验室
Advanced Compiler

嘉宾：柴赟达

循环合并

- **基础概念**

循环合并，Loop Fusion，是将具有相同迭代空间的两个循环合成一个循环的过程，属于语句层次的循环变换。

for (i = 0; i < N; i++)
    x[i] = a[i] + b[i];
for (i = 0; i < N; i++)
    y[i] = a[i] - b[i];
	for (i = 0; i < N; i++) {
	    x[i] = a[i] + b[i];
	    y[i] = a[i] - b[i];
	}

先进编译实验室 Advanced Compiler
先进编译实验室 Advanced Compiler

- **优点：**

① 减小循环的迭代开销

② 增强数据重用，寄存器重用

③ 减小并行化的启动开销，消除合并前多个循环间的线程同步开销

④ 增加循环优化的范围

循环合并

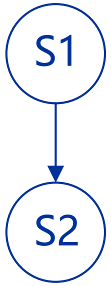
- **循环合并的合法性**

① 不能违反原来的依赖关系图。如果两个循环之间存在一条循环无关的依赖路径，这条路径包含一个未与它们合并的循环语句，则它们不能被合并。

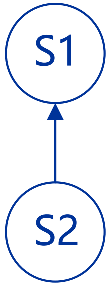
② 不能产生新的依赖。如果两个循环之间存在一个阻止合并的依赖，则它们不能被合并。

```
for (i = 1; i < N; i++)
    A[i] = B[i] + C;//S1语句
for (i = 1; i < N; i++)
    D[i] = A[i + 1] + E;//S2语句
```

```
for (i = 1; i < N; i++) {
    A[i] = B[i] + C;//S1语句
    D[i] = A[i + 1] + E;//S2语句
}
```



循环无关依赖



循环携带反依赖

先进编译实验室 Advanced Compiler

循环合并

- **循环合并的有利性**

循环合并的一个重要应用场景为并行化，但并不是所有循环合并都可以给并行化带来收益，有以下两种情况：

① 分离限制：当一个并行循环和一个串行循环合并时，结果必然是串行执行的。

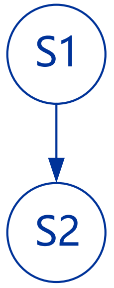
```
for (i = 1; i < N; i++)
    A[i] = B[i] + 1;//S1
for (i = 1; i < N; i++)
    C[i] = A[i] + C[i - 1];//S2
```

```
for (i = 1; i < N; i++){
    A[i] = B[i] + 1;//S1
    C[i] = A[i] + C[i - 1];//S2
}
```

② 阻止并行性的依赖限制，当两个都可并行的循环存在一个阻止并行的依赖，进行循环合并后该依赖被合并后的循环携带。

```
for (i = 1; i < N; i++)
    A[i + 1] = B[i] + C;//S1
for (i = 1; i < N; i++)
    D[i] = A[i] + E;//S2
```

```
for (i = 1; i < N; i++){
    A[i + 1] = B[i] + C;//S1
    D[i] = A[i] + E;//S2
}
```



循环携带依赖

先进编译实验室
Advanced Compiler

循环合并

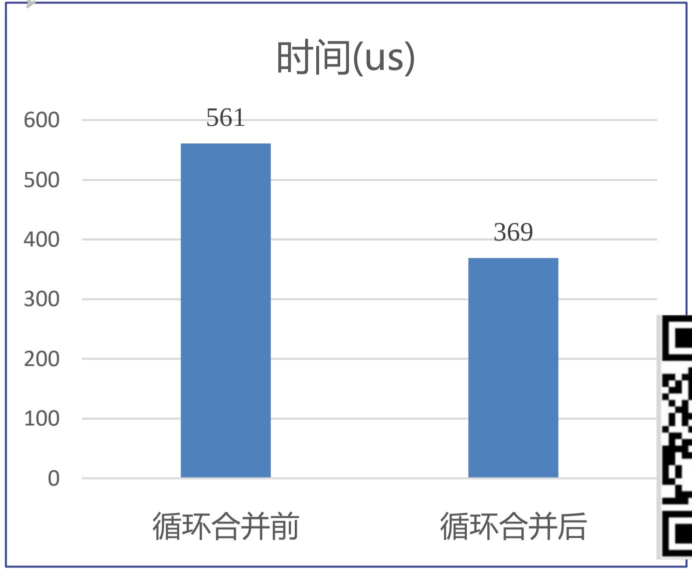
优化效果

测试环境：Hygon C86 7185 32-core Processor；x86_64；

编译器版本：llvm-13

测试例：

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#define N 51200
int main() {
  int i, a[N], b[N], x[N], y[N];
  struct timeval  time_start, time_end;
  for (i = 0; i < N; i++) {
    a[i] = rand()%100;
    b[i] = rand()%100;
  }
  gettimeofday(&time_start, NULL);
  for (i = 0; i < N; i++)
    x[i] = a[i] + b[i];
  for (i = 0; i < N; i++)
    y[i    [i] - b[i];
             y(&time_end, NULL);
             l used time %ld us\n", time_end.tv_usec - time_start.tv_usec);
             ", x[4]);
  p          d\n", y[3]);
```



先进编译实验室
Advanced Compiler
先进编译实验室
Advanced Compiler
先进编译实验室
Advanced Compiler

循环合并

编译器中的循环合并

LLVM中进行循环合并需要满足以下条件：

① 两个循环必须相邻，即两个循环之间不能有语句。

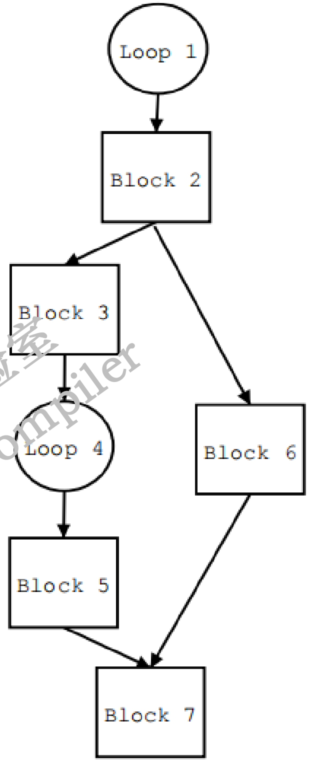
② 两个循环必须有相同的迭代次数。

③ 循环必须是等价的控制流，如果一个循环执行，另一个循环也保证执行。

④ 循环之间不能有负距离依赖关系，比如两个循环，当第二个循环的迭代次数为m时，使用第一个循环在未来m+n次迭代时计算的值(其中n > 0)。

先进编译实验室
Advanced Compiler

```
for (i = 1; i < N; i++)
    A[i] = B[i] + C;//L_j
for (i = 1; i < N; i++)
    D[i] = A[i + 4] + E;//L_k
```



先进编译实验室
Advanced Compiler

选项	功能
-fproactive-loop-fusion-analysis	打开循环合并分析遍
-fproactive-loop-fusion	打开循环合并优化遍
-loop-fusion	通过opt工具对中间码进行循环合并优化

循环合并

编译器中的循环合并

opt -S -loop-fusion file.ll

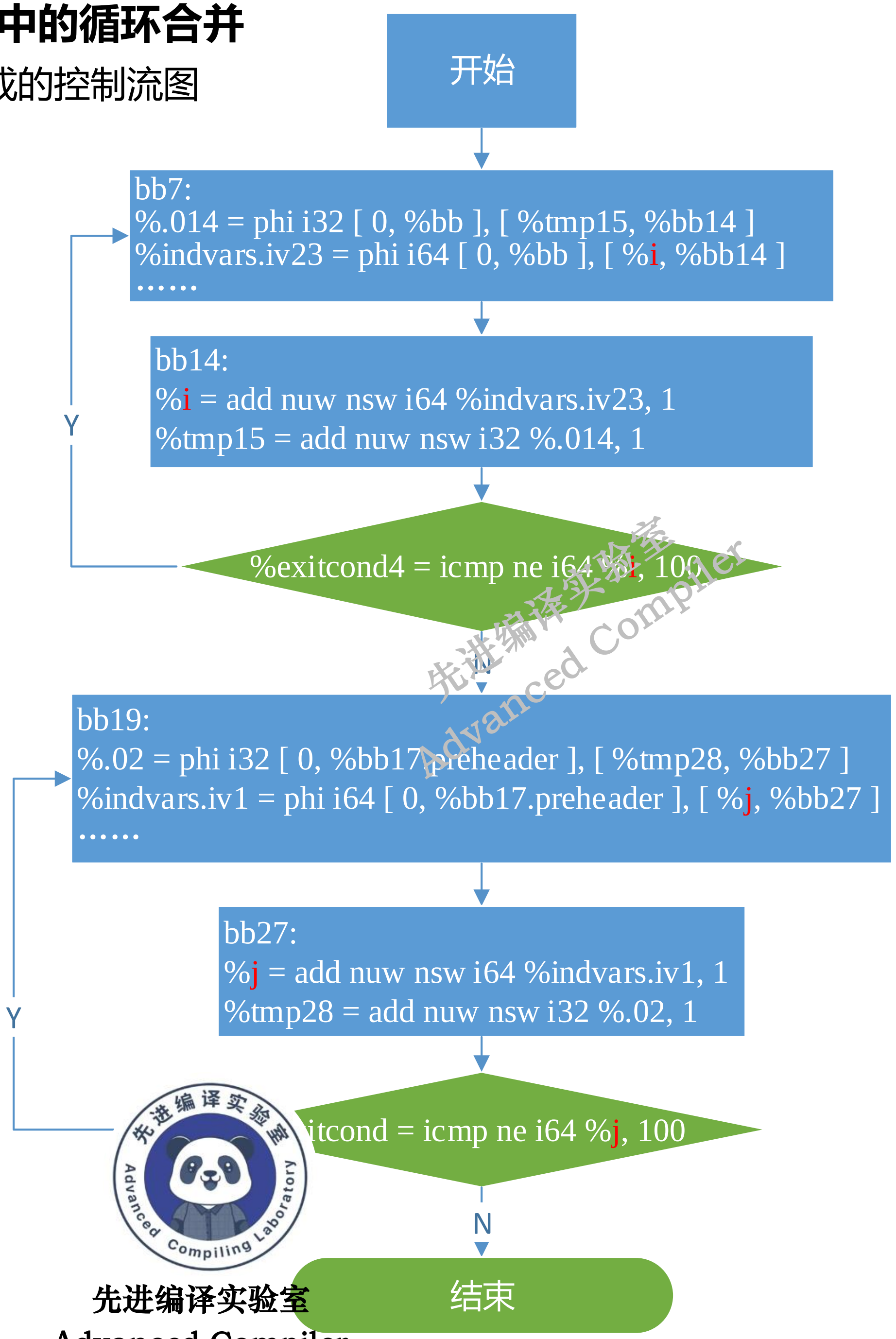
```
bb7:                                    ; preds = %bb, %bb14
  %.014 = phi i32 [ 0, %bb ], [ %tmp15, %bb14 ]
  %indvars.iv23 = phi i64 [ 0, %bb ], [ %indvars.iv.next3, %bb14 ]
  %tmp = add nsw i32 %.014, -3
  %tmp8 = add nuw nsw i64 %indvars.iv23, 3
……
  %tmp13 = getelementptr inbounds i32, i32* %arg, i64 %indvars.iv23
  store i32 %tmp12, i32* %tmp13, align 4
  br label %bb14
bb14:                                   ; preds = %bb7
  %indvars.iv.next3 = add nuw nsw i64 %indvars.iv23, 1
  %tmp15 = add nuw nsw i32 %.014, 1
  %exitcond4 = icmp ne i64 %indvars.iv.next3, 100
  br i1 %exitcond4, label %bb7, label %bb17.preheader
bb17.preheader:                         ; preds = %bb14
  br label %bb19
bb19:                                   ; preds = %bb17.preheader, %bb27
  %.02 = phi i32 [ 0, %bb17.preheader ], [ %tmp28, %bb27 ]
  %indvars.iv1 = phi i64 [ 0, %bb17.preheader ], [ %indvars.iv.next, %bb27 ]
…….
 %tmp26 = getelementptr inbounds [1024 x i32], [1024 x i32]* @B, i64 0, i64
%indvars.iv1
  store i32 %tmp25, i… align 4
  br label %bb27
bb27:                                   …reds = %bb19
  %indvars.iv.next = … 64 %indvars.iv1, 1
  %tmp28 = add nuw ns… .02, 1
  %exitcond = icmp… s.iv.next, 100
  br i1 %exitcond… label %bb19, label %bb18
```

```
bb7:                                    ; preds = %bb27, %bb
  %.014 = phi i32 [ 0, %bb ], [ %tmp15, %bb27 ]
  %indvars.iv23 = phi i64 [ 0, %bb ], [ %indvars.iv.next3, %bb27 ]
  %.02 = phi i32 [ 0, %bb ], [ %tmp28, %bb27 ]
  %indvars.iv1 = phi i64 [ 0, %bb ], [ %indvars.iv.next, %bb27 ]
  %tmp = add nsw i32 %.014, -3
  %tmp8 = add nuw nsw i64 %indvars.iv23, 3
…….
  %tmp13 = getelementptr inbounds i32, i32* %arg, i64 %indvars.iv23
  store i32 %tmp12, i32* %tmp13, align 4
  br label %bb14
bb14:                                   ; preds = %bb7
  %indvars.iv.next3 = add nuw nsw i64 %indvars.iv23, 1
  %tmp15 = add nuw nsw i32 %.014, 1
  %exitcond4 = icmp ne i64 %indvars.iv.next3, 100
  br i1 %exitcond4, label %bb19, label %bb19
bb19:                                   ; preds = %bb14, %bb14
……
  %tmp26 = getelementptr inbounds [1024 x i32], [1024 x i32]* @B, i64 0, i64 %indvars.iv1
  store i32 %tmp25, i32* %tmp26, align 4
  br label %bb27
bb27:                                   ; preds = %bb19
  %indvars.iv.next = add nuw nsw i64 %indvars.iv1, 1
  %tmp28 = add nuw nsw i32 %.02, 1
  %exitcond = icmp ne i64 %indvars.iv.next, 100
  br i1 %exitcond, label %bb7, label %bb18
```

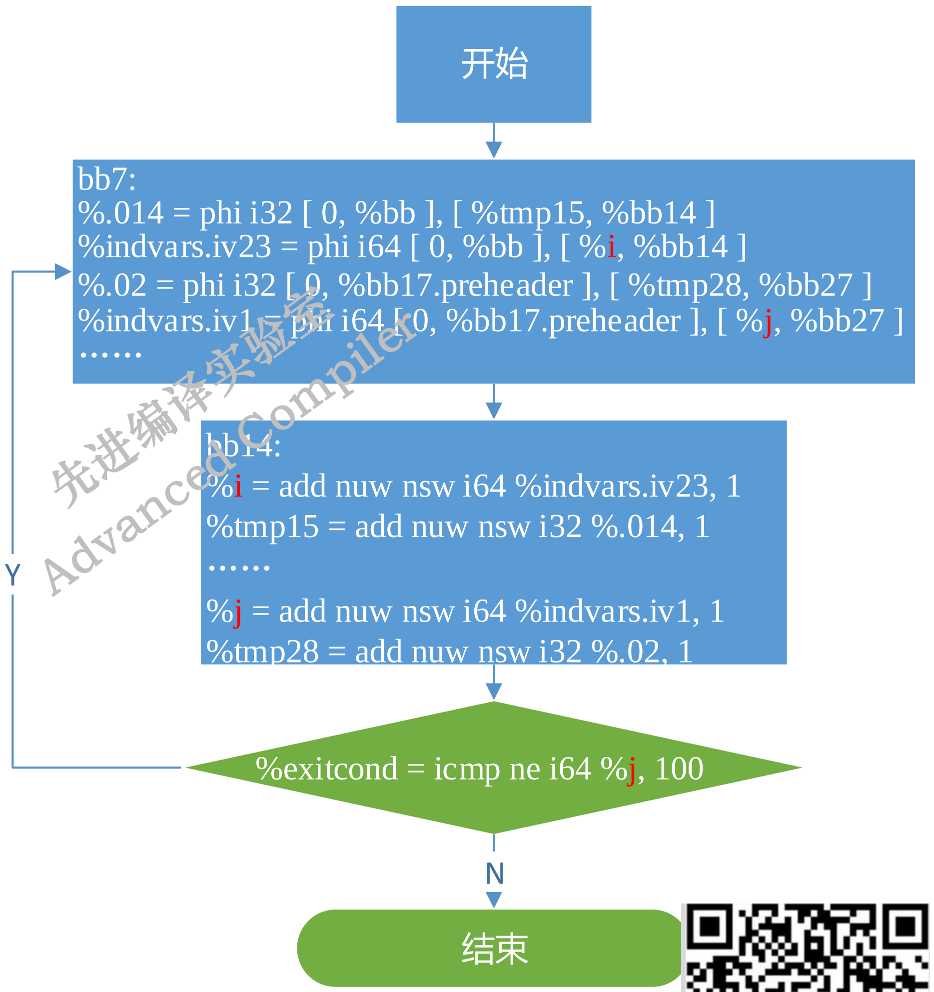
先进编译实验室 Advanced Compiler

编译器中的循环合并

对应生成的控制流图



opt -S -loop-fusion file.ll



先进编译实验室
Advanced Compiler
先进编译实验室 Advanced Compiler

分享完毕，感谢聆听！

参考文献：

[1] Kai Nacke.Learn LLVM 12, A beginner’s guide to learning LLVM compiler tools and core libraries with C++[M]. Packt Publishing Ltd.,2021

[2] Optimizing Compilers for Modern Architectures: A Dependence-Based Approach [Book Review][J]. Computer,2002,35(4).

[3] 胡伟方,陈云,李颖颖,商建东.基于数据重用分析的多面体循环合并策略[J].计算机科学,2021,48(12):49-58.

先进编译实验室
Advanced Compiler