

HyBF: A Hybrid Branch Fusion Strategy for Code Size Reduction

R. Rocha, C. Saumya, K. Sundararajah, P. Petoumenos, M. Kulkarni, M. O'Boyle

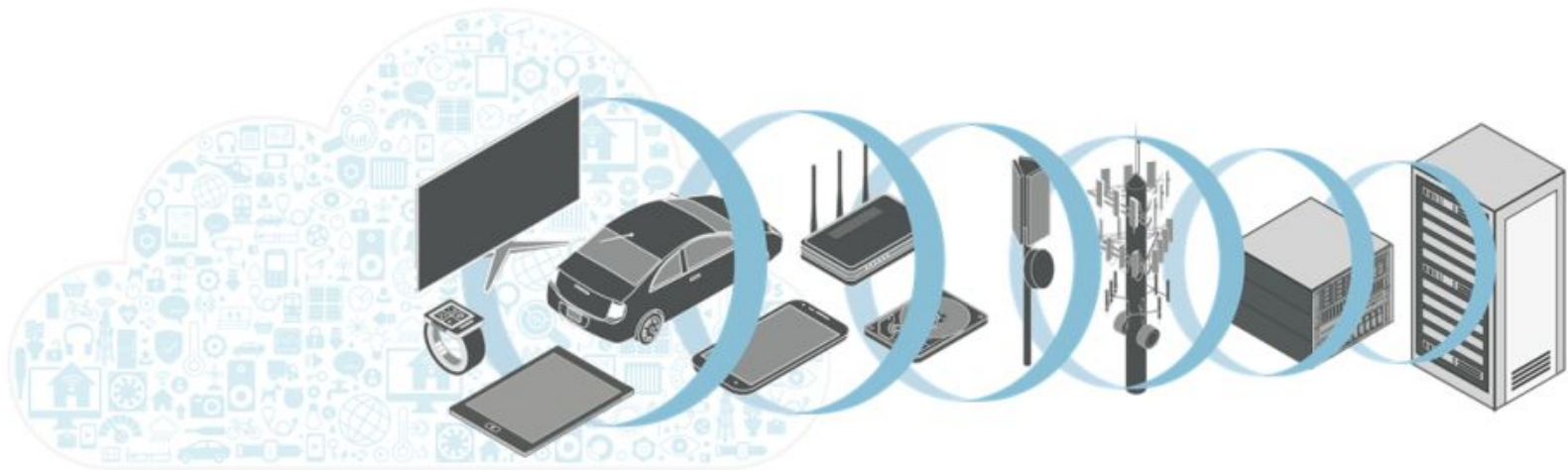


HyBF: A Hybrid Branch Fusion Strategy for Code Size Reduction

R. Rocha, C. Saumya, K. Sundararajah, P. Petoumenos, M. Kulkarni, M. O'Boyle

Code Size Matters

Code size is important in many domains -- Large is relative to the constraints.
Compilers need powerful optimizations for code size.



What is Branch Fusion?

```
if (atomic_dec_test(&rcus.bcpucount)) {  
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);  
    complete(&rcus.bcompletion);  
} else {  
    rcu_btrace(TPS("CB"), -1, rcus.bseq);  
}
```

What is Branch Fusion?

Identify similarities between the *then* and *else* paths in conditional branches.

```
if (atomic_dec_test(&rcus.bcpucount)) {  
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);  
    complete(&rcus.bcompletion);  
} else {  
    rcu_btrace(TPS("CB"), -1, rcus.bseq);  
}
```

What is Branch Fusion?

Identify similarities between the *then* and *else* paths in conditional branches.

```
if (atomic_dec_test(&rcus.bcpucount)) {  
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);  
    complete(&rcus.bcompletion);  
} else {  
    rcu_btrace(TPS("CB"), -1, rcus.bseq);  
}
```

What is Branch Fusion?

Merge similar code from the *then* and *else* paths of the branch.

```
if (atomic_dec_test(&rcus.bcpucount)) {    cond = atomic_dec_test(&rcus.bcpucount)
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);
    complete(&rcus.bcompletion);
} else {
    rcu_btrace(TPS("CB"), -1, rcus.bseq);
}
```

What is Branch Fusion?

Merge similar code from the *then* and *else* paths of the branch.

```
if (atomic_dec_test(&rcus.bcpucount)) {    cond = atomic_dec_test(&rcus.bcpucount)
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);
    complete(&rcus.bcompletion);
} else {
    rcu_btrace(TPS("CB"), -1, rcus.bseq);
}
```


What is Branch Fusion?

Merge similar code from the *then* and *else* paths of the branch.

```
if (atomic_dec_test(&rcus.bcpucount)) {    cond = atomic_dec_test(&rcus.bcpucount)
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);    mval = cond ? "LastCB" : "CB";
    complete(&rcus.bcompletion);    rcu_btrace(TPS(mval), -1, rcus.bseq);
} else {
    rcu_btrace(TPS("CB"), -1, rcus.bseq);
}
```

What is Branch Fusion?

Merge similar code from the *then* and *else* paths of the branch.

```
if (atomic_dec_test(&rcus.bcpucount)) {  
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);  
    complete(&rcus.bcompletion);  
} else {  
    rcu_btrace(TPS("CB"), -1, rcus.bseq);  
}  
  
cond = atomic_dec_test(&rcus.bcpucount)  
mval = cond ? "LastCB" : "CB";  
rcu_btrace(TPS(mval), -1, rcus.bseq);  
if (cond) {  
    complete(&rcus.bcompletion);  
}
```

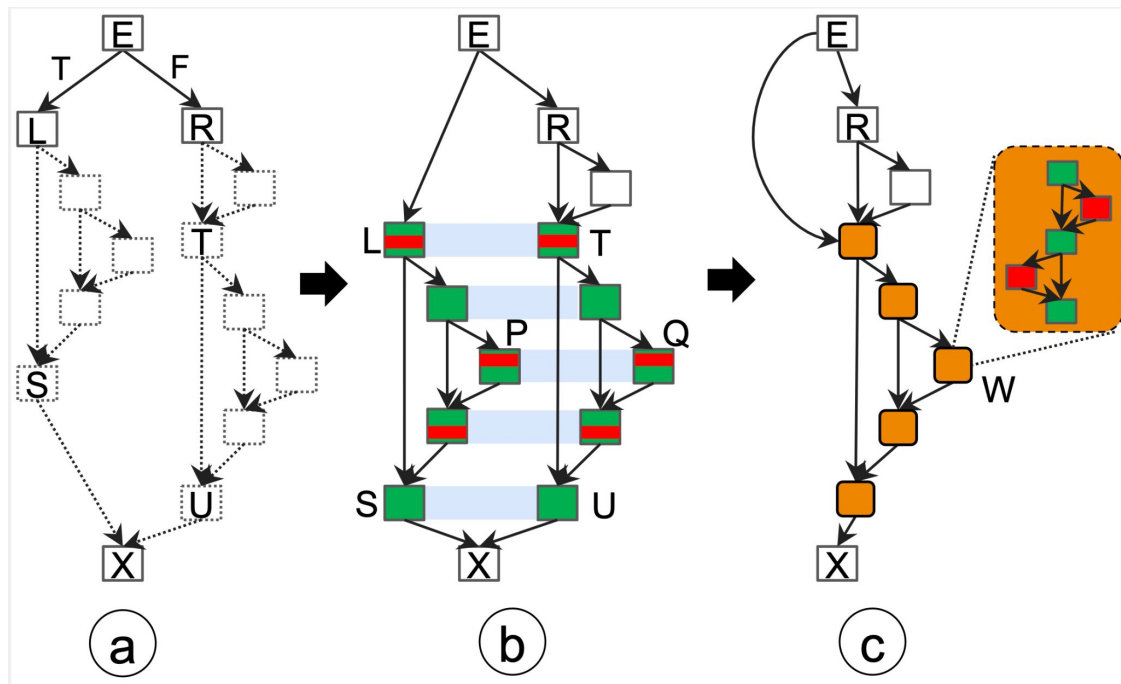
Main Insight: Branch Fusion Reduces Code Size

```
if (atomic_dec_test(&rcus.bcpucount)) {  
    rcu_btrace(TPS("LastCB"), -1, rcus.bseq);  
    complete(&rcus.bcompletion);  
} else {  
    rcu_btrace(TPS("CB"), -1, rcus.bseq);  
}  
  
cond = atomic_dec_test(&rcus.bcpucount)  
mval = cond ? "LastCB" : "CB";  
rcu_btrace(TPS(mval), -1, rcus.bseq);  
if (cond) {  
    complete(&rcus.bcompletion);  
}
```

18 bytes (11%) Reduction

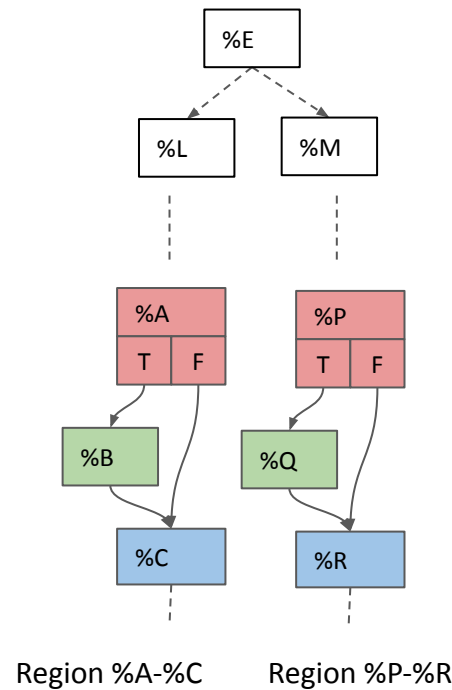
CFM-CS

Improved DARM to work on complex SESE regions in generic (non-GPU) code.



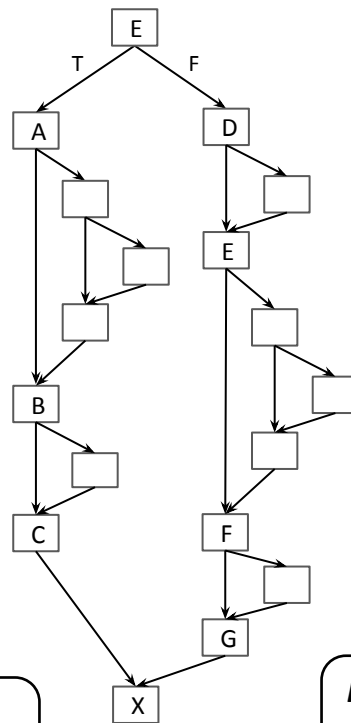
Meldable Regions

- Two SESE regions can be ***melded*** if,
 - Dominated by a conditional branch
 - No path exists that goes through both the SESE regions
 - Entry blocks of the regions must post-dominate either the left or right successor of the conditional branch
 - They are ***isomorphic*** (structural similarity)

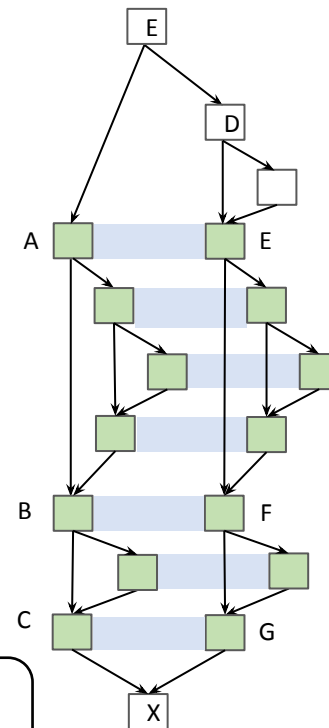


Region Alignment

- Multiple isomorphic regions in if and else paths?
- Regions are aligned based on **Melding Profitability**
- **Melding Profitability** : metric that measures the similarity of two regions base on instruction frequencies

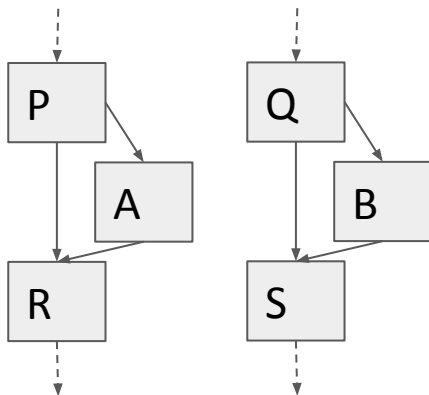


Left regions : A-B, B-C
Right Regions : D-E, E-F, F-G

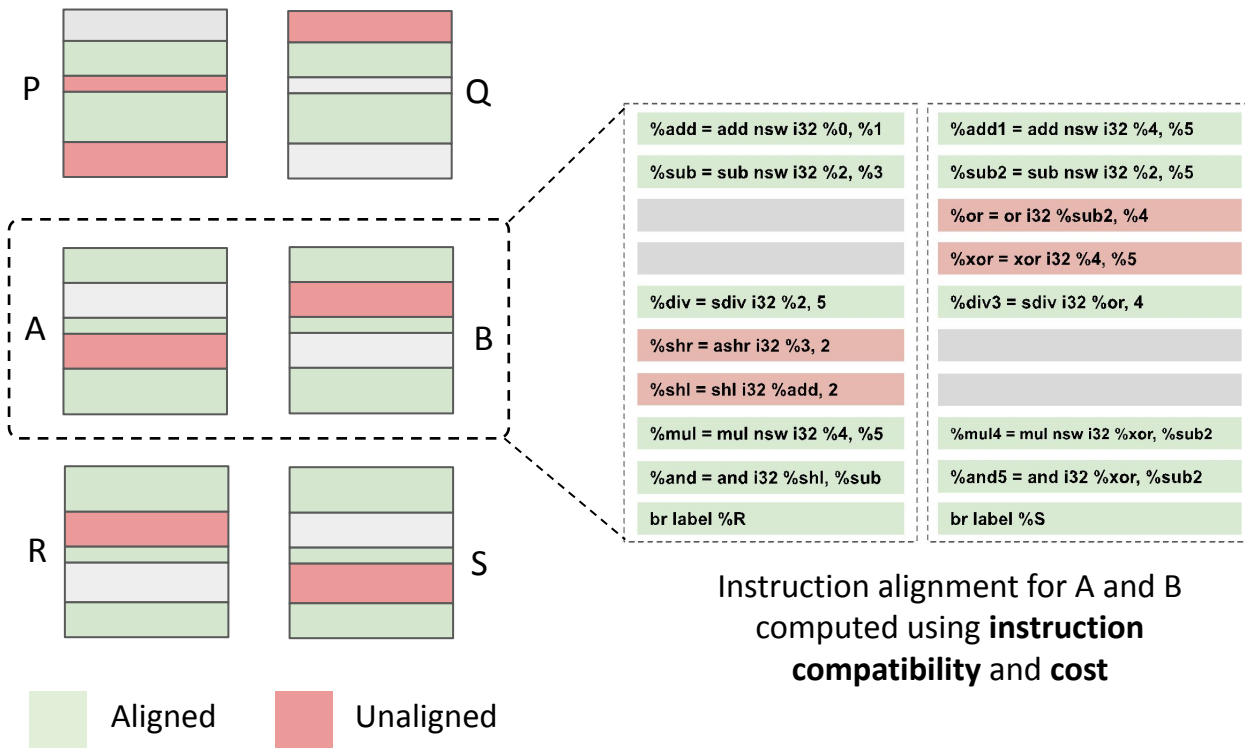


Region Alignment :
A-B with E-F
B-C with F-G

Instruction Alignment

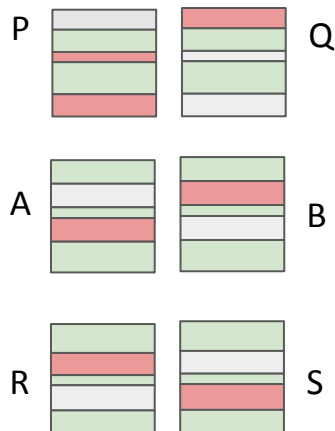
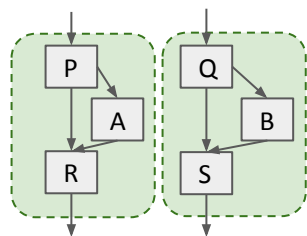


Aligned region pair

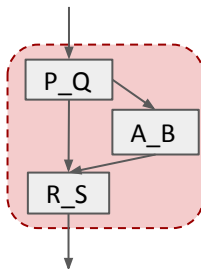


Instruction alignment for A and B
computed using **instruction
compatibility** and **cost**

Code Generation



Generated melded
control-flow



Generated melded
instructions

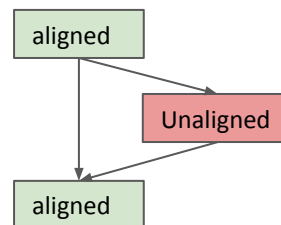
Aligned instruction pair

```
%add = add nsw i32 %0, %1
```

```
%add1 = add nsw i32 %4, %5
```

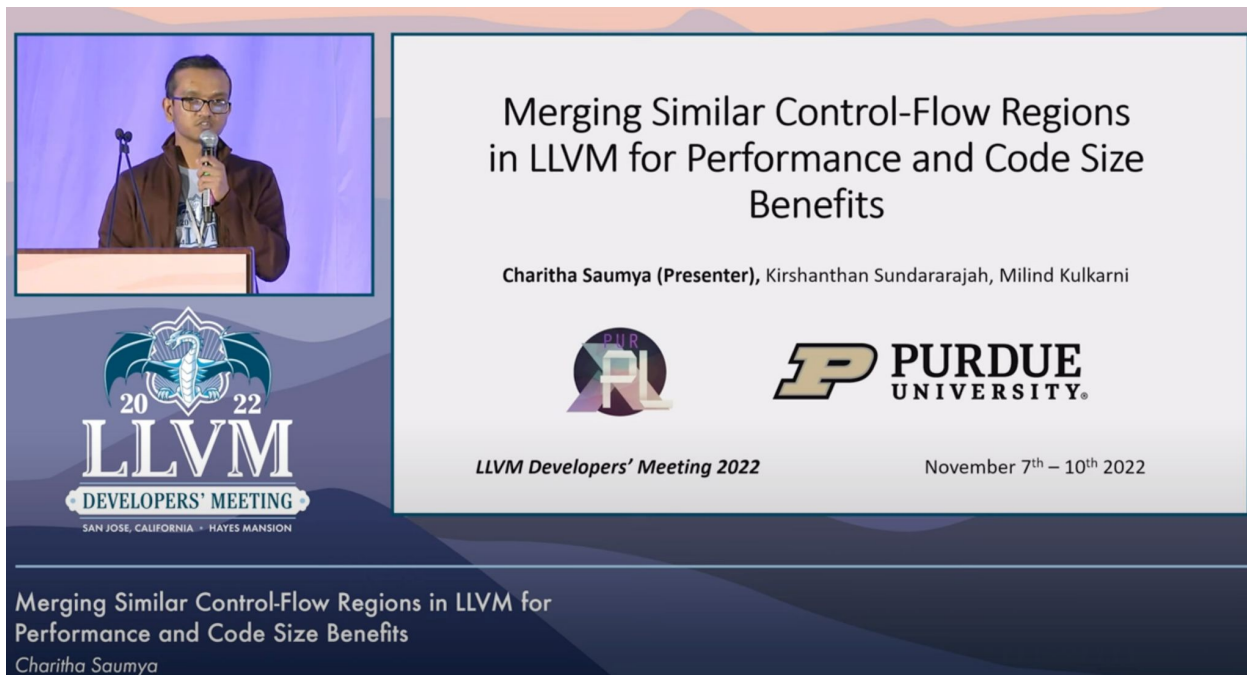
Generated code

```
%sel1 = select i1 %cmp, i32 %0, i32 %4
%sel2 = select i1 %cmp, i32 %1, i32 %5
%6 = add nsw i32 %sel1, %sel2
```



Unaligned instructions are
executed conditionally



CFM-CS: Deep Dive Talk



The screenshot shows a presentation slide from the 2022 LLVM Developers' Meeting. On the left, there is a video inset of a man with glasses speaking at a podium. The main slide has a light blue background with a title, presenter information, logos, and dates. At the bottom of the slide, there is a dark blue banner with the title and presenter name repeated.

Merging Similar Control-Flow Regions in LLVM for Performance and Code Size Benefits

Charitha Saumya (Presenter), Kirshanthan Sundararajah, Milind Kulkarni

LLVM Developers' Meeting 2022 November 7th – 10th 2022

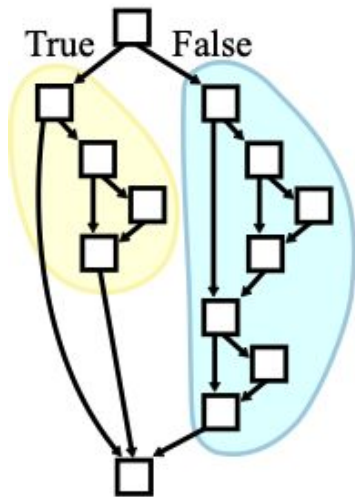
2022 LLVM DEVELOPERS' MEETING
SAN JOSE, CALIFORNIA – HAYES MANSION

Merging Similar Control-Flow Regions in LLVM for Performance and Code Size Benefits
Charitha Saumya

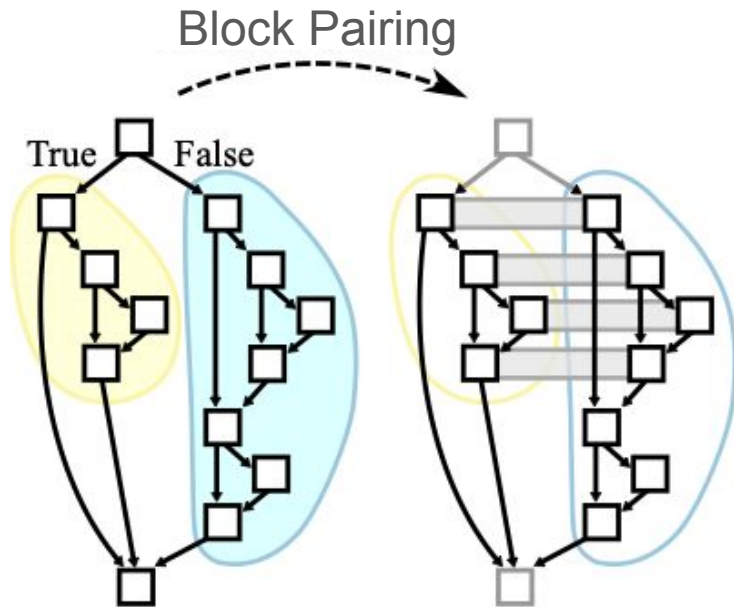
<https://www.youtube.com/watch?v=iGbdcltU0F8>

2022 LLVM Dev Mtg: Merging Similar Control-Flow Regions

Branch Fusion on SEME Regions

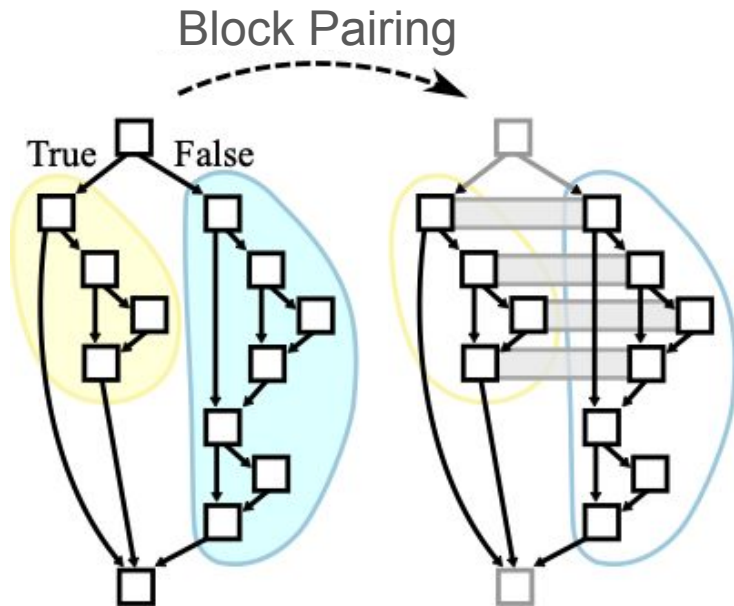


Branch Fusion on SEME Regions



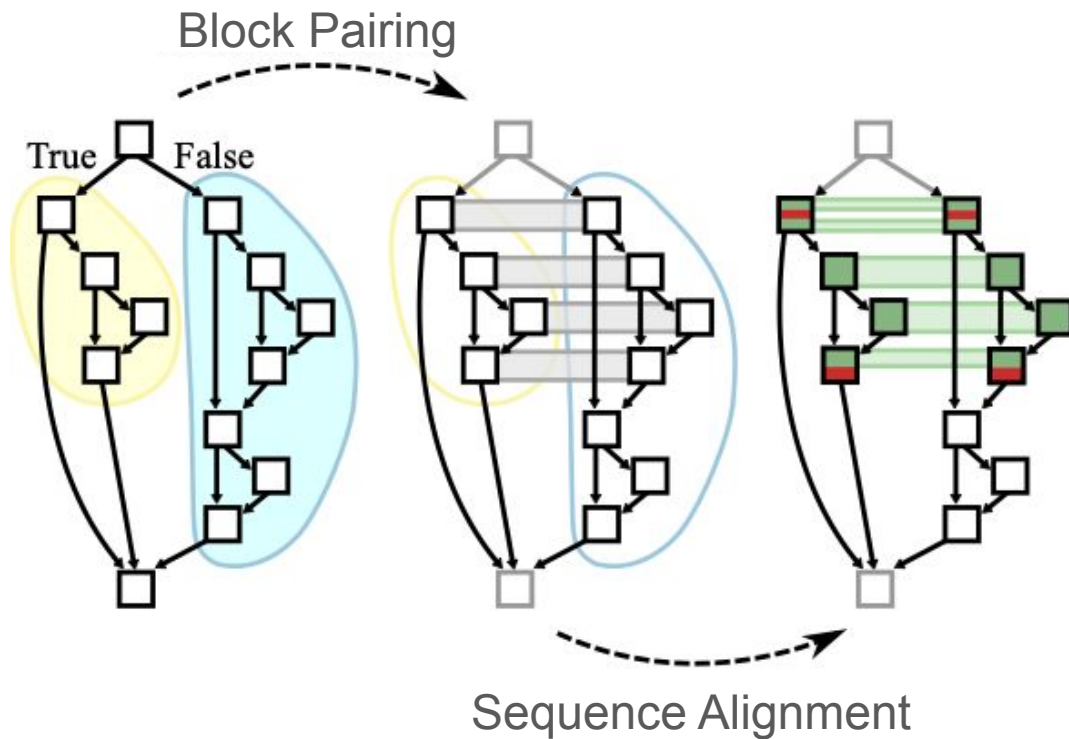
Blocks are paired independently, based only on their similarity.

Branch Fusion on SEME Regions

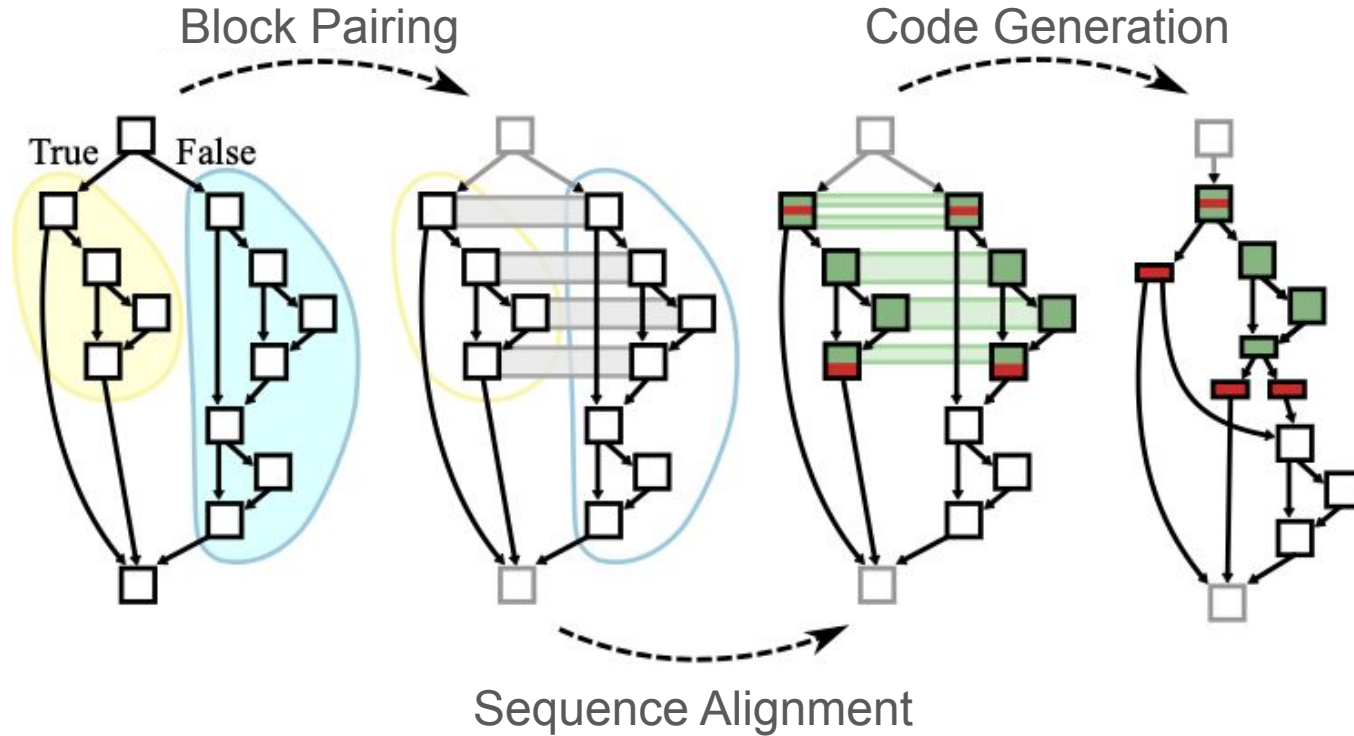


Blocks are paired independently, based only on their similarity.

Branch Fusion on SEME Regions



Branch Fusion on SEME Regions



Pairing Similar Blocks

Given two SEME regions

Region 1



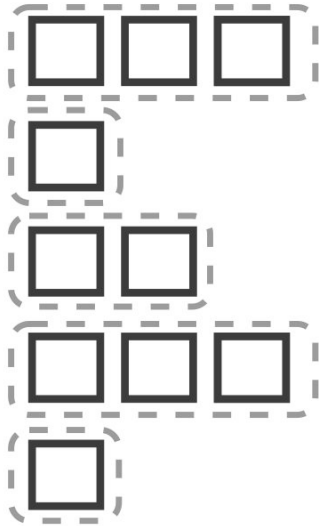
Region 2



Pairing Similar Blocks

Group blocks by their size

Region 1



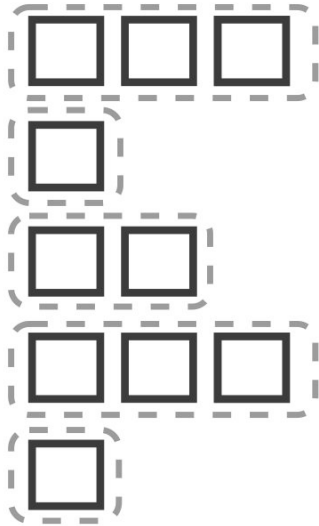
Region 2



Pairing Similar Blocks

Minimize fingerprint distance

Region 1



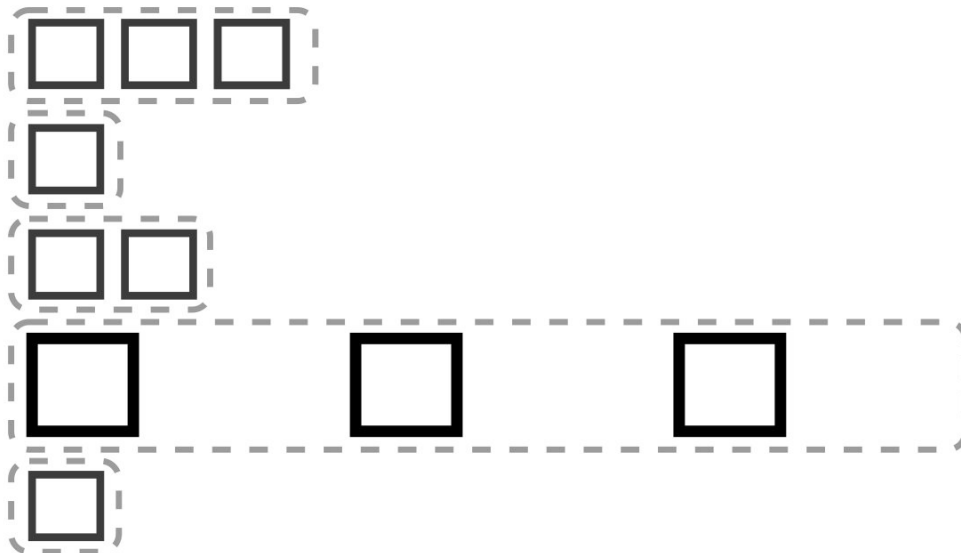
Region 2



Pairing Similar Blocks

Minimize fingerprint distance

Region 1

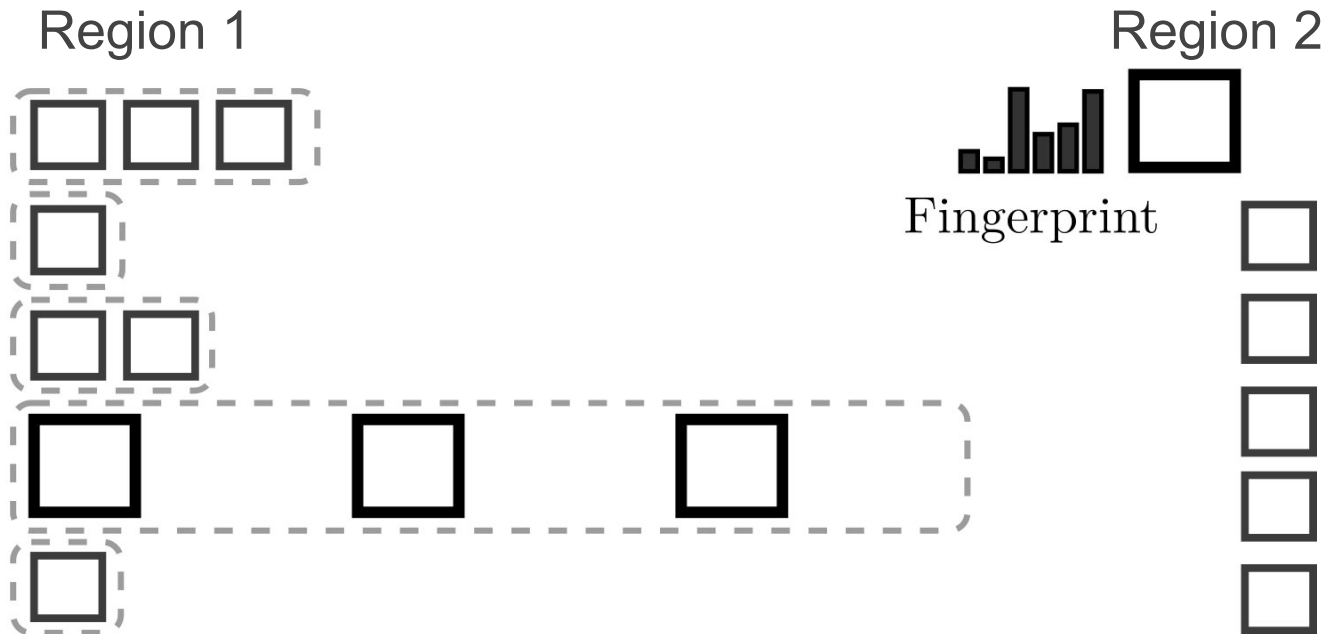


Region 2



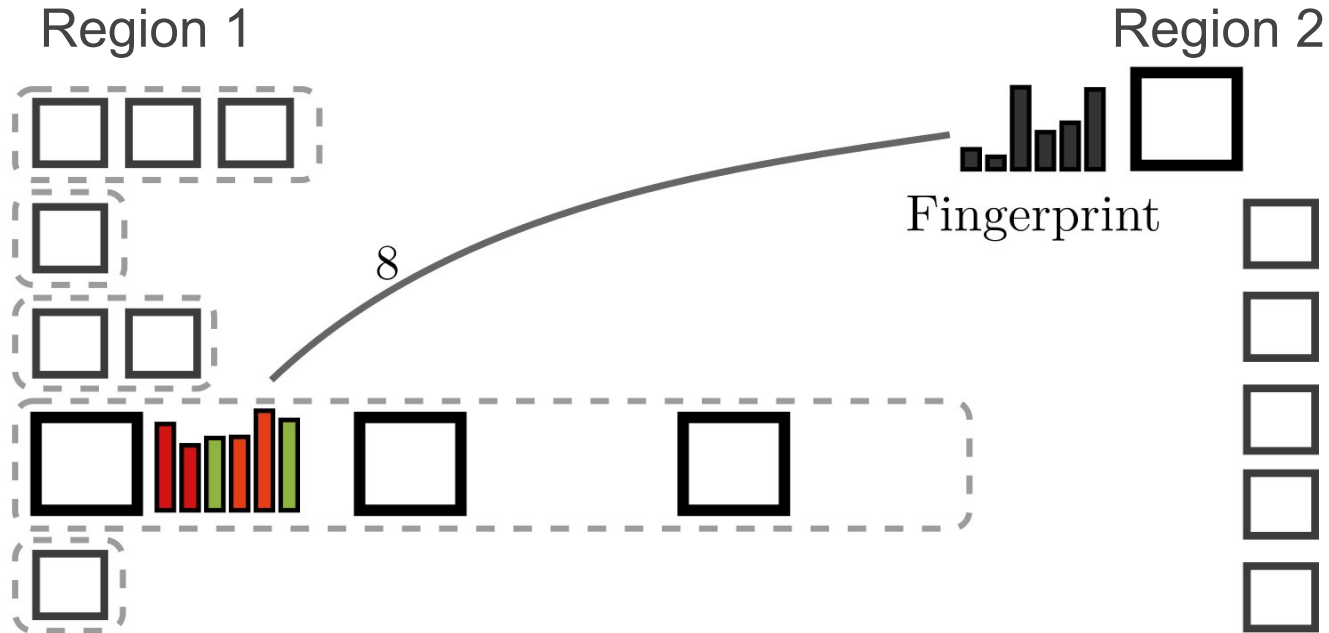
Pairing Similar Blocks

Minimize fingerprint distance



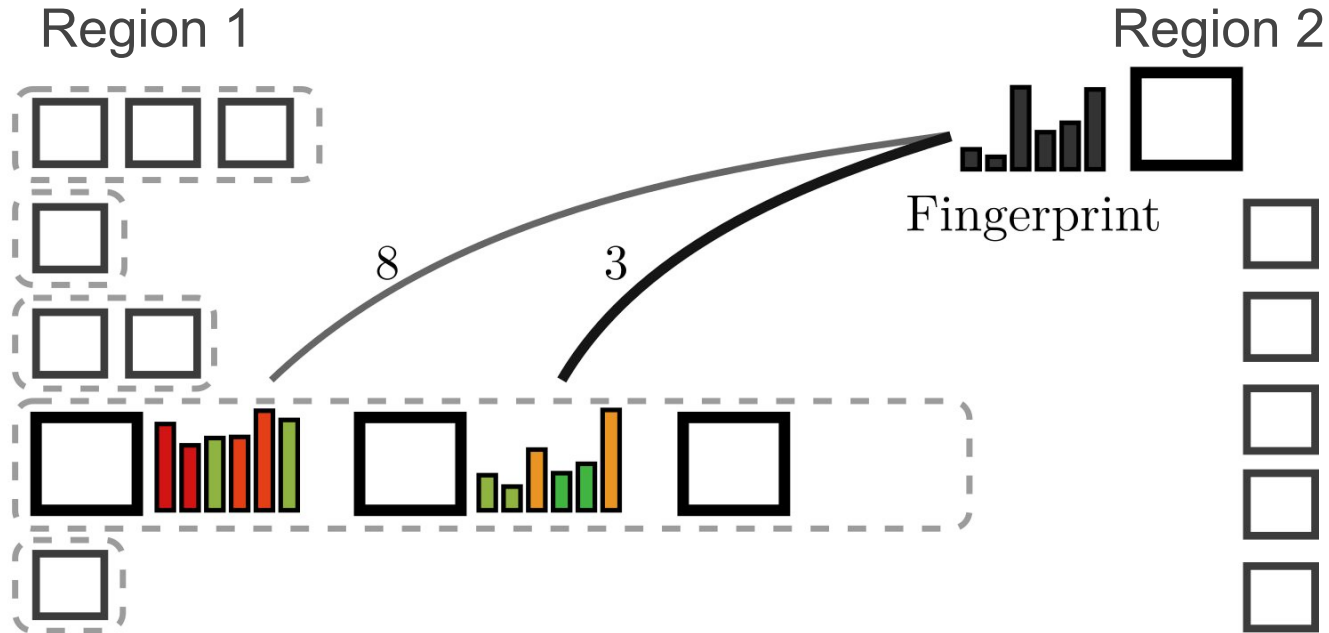
Pairing Similar Blocks

Minimize fingerprint distance



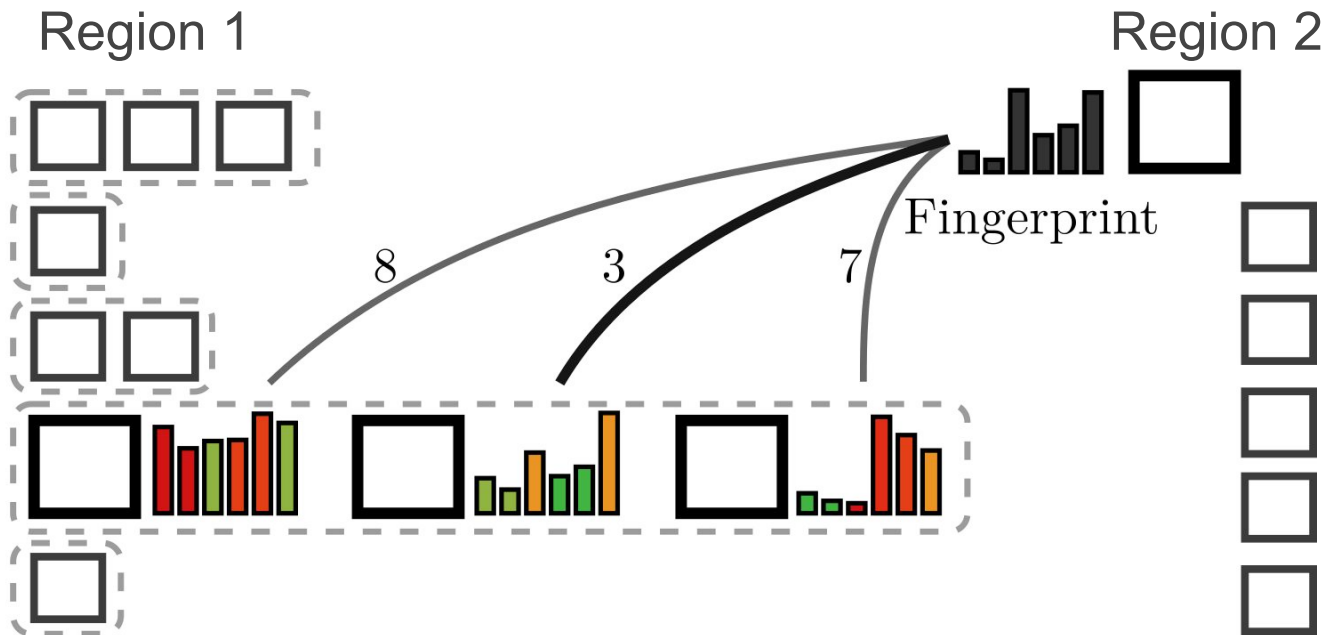
Pairing Similar Blocks

Minimize fingerprint distance



Pairing Similar Blocks

Minimize fingerprint distance



Pairwise Alignment of Paired Blocks

Match only corresponding instructions

<code>%sw.bb</code>	<code>%entry</code>
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>
<code>br %v4, L_{b3}, L_{b2}</code>	<code>br %x4, L_{b3}, L_{b2}</code>

Pairwise Alignment of Paired Blocks

Match only corresponding instructions

<code>%sw.bb</code>	<code>%entry</code>
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>
<code>br %v4, L_{b3}, L_{b2}</code>	<code>br %x4, L_{b3}, L_{b2}</code>

Pairwise Alignment of Paired Blocks

Match only corresponding instructions

<code>%sw.bb</code>	<code>%entry</code>
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>

Pairwise Alignment of Paired Blocks

Match only corresponding instructions

<code>%sw.bb</code>	<code>%entry</code>
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>

Pairwise Alignment of Paired Blocks

Match only corresponding instructions

<code>%sw.bb</code>	<code>%entry</code>
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>
<code>br %v4, L_{b3}, L_{b2}</code>	<code>br %x4, L_{b3}, L_{b2}</code>

Pairwise Alignment of Paired Blocks

Match only corresponding instructions

<code>%sw.bb</code>	<code>%entry</code>
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>	0
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>	

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>	0
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	2
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>	

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>	0
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	2
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>	

} +1

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>	0
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	2
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	2
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>	

) +1

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>	0
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	2
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	2
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	1
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>	

) +1

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>	
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	0
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	2
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	2
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	1
<code>br %v4, L_{b3}, L_{b2}</code>	<code>br %x4, L_{b3}, L_{b2}</code>	

Annotations:
 - A bracket on the right side of the first two rows (0 and 2) is labeled $+1$.
 - A bracket on the right side of the next two rows (2 and 1) is labeled $+2$.

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>		
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	0	} +1
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	2	
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	2	} +2
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	1	
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>	1	

Pairwise Alignment of Paired Blocks

Estimate size of merged block

<code>%sw.bb</code>	<code>%entry</code>	0	} +1
<code>%v1 = gep %this, 0, 5</code>	<code>%x1 = alloca</code>	2	
<code>%v2 = bitcast %v1</code>	<code>%x2 = gep %this, 0, 1</code>	2	} +2
<code>%v3 = load %v2</code>	<code>%x3 = load %x2</code>	1	
<code>%v4 = icmp eq %v3, 0</code>	<code>%x4 = icmp eq %x3, 73</code>	1	
<code>br %v4, Lb3, Lb2</code>	<code>br %x4, Lb3, Lb2</code>	1	

Merged Cost: 10 ✓

Main Differences Between CFM-CS and SEME-Fusion

Structural similarity

CFM-CS only works on branches with isomorphic SESE subregions.

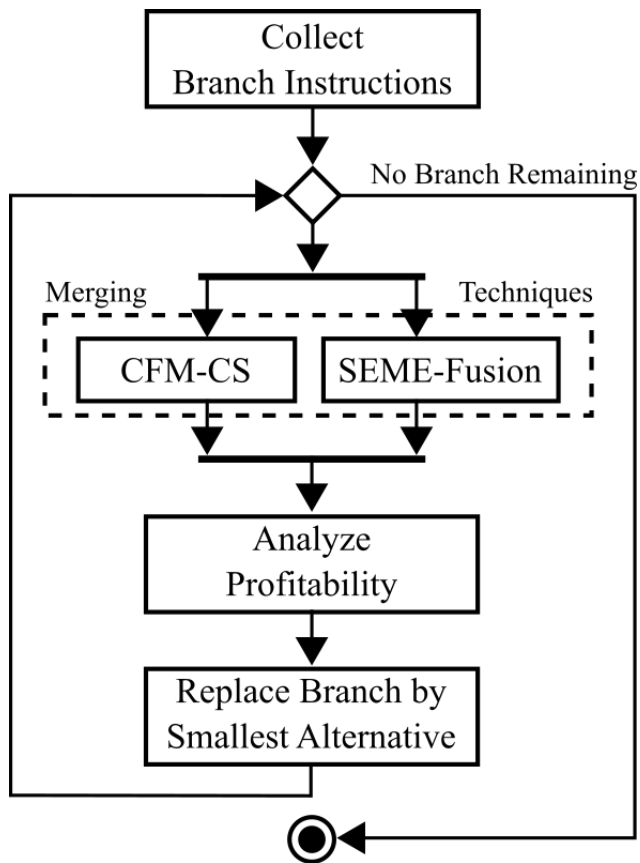
SEME-Fusion can merge any conditional branch - no structural similarity needed.

Block Paring

CFM-CS pairs the corresponding blocks in the isomorphic SESE subregions

SEME-Fusion pairs blocks based on fingerprint distance

HyBF: The Best of CFM-CS and SEME-Fusion



CFM-CS and SEME-Fusion compete,
the best result wins!

Evaluation Setup

Implemented in Clang/LLVM.

Benchmarks optimized with -Oz for code size.

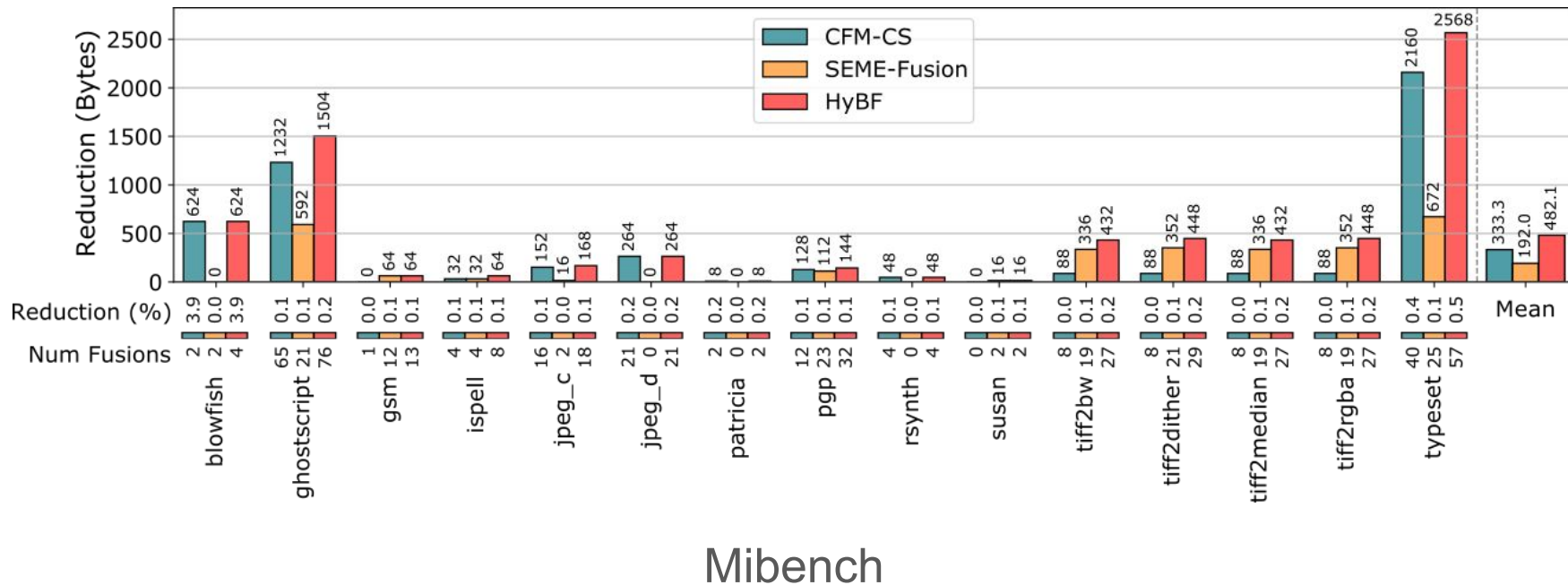
CFM-CS

SEME-Fusion

HyBF: Combines both CFM-CS and SEME-Fusion

Size Reduction

HyBF achieves the best reduction.

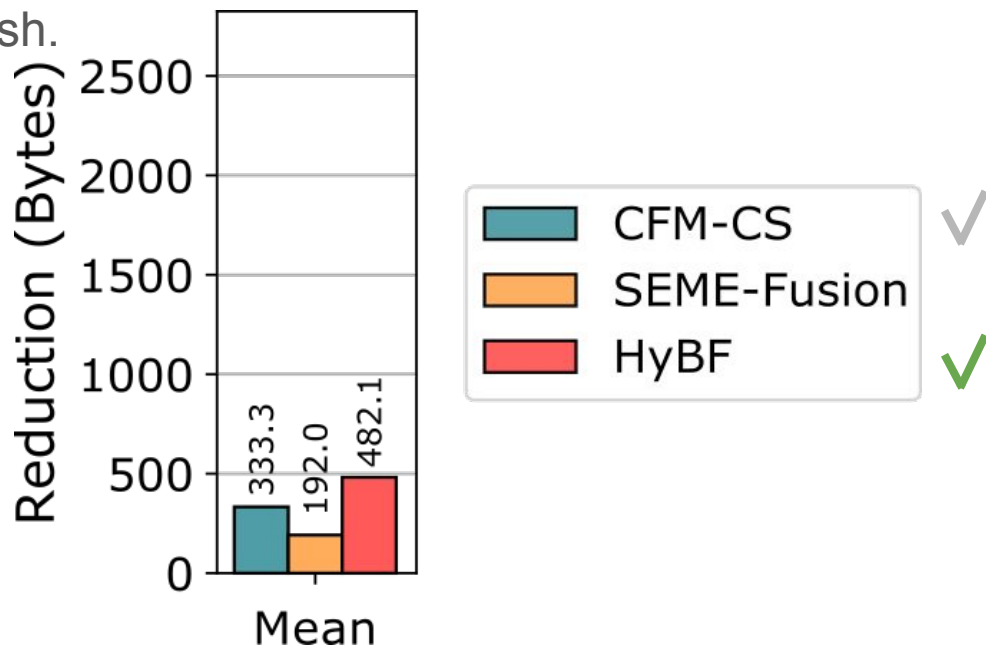


Size Reduction

HyBF achieves the best reduction.

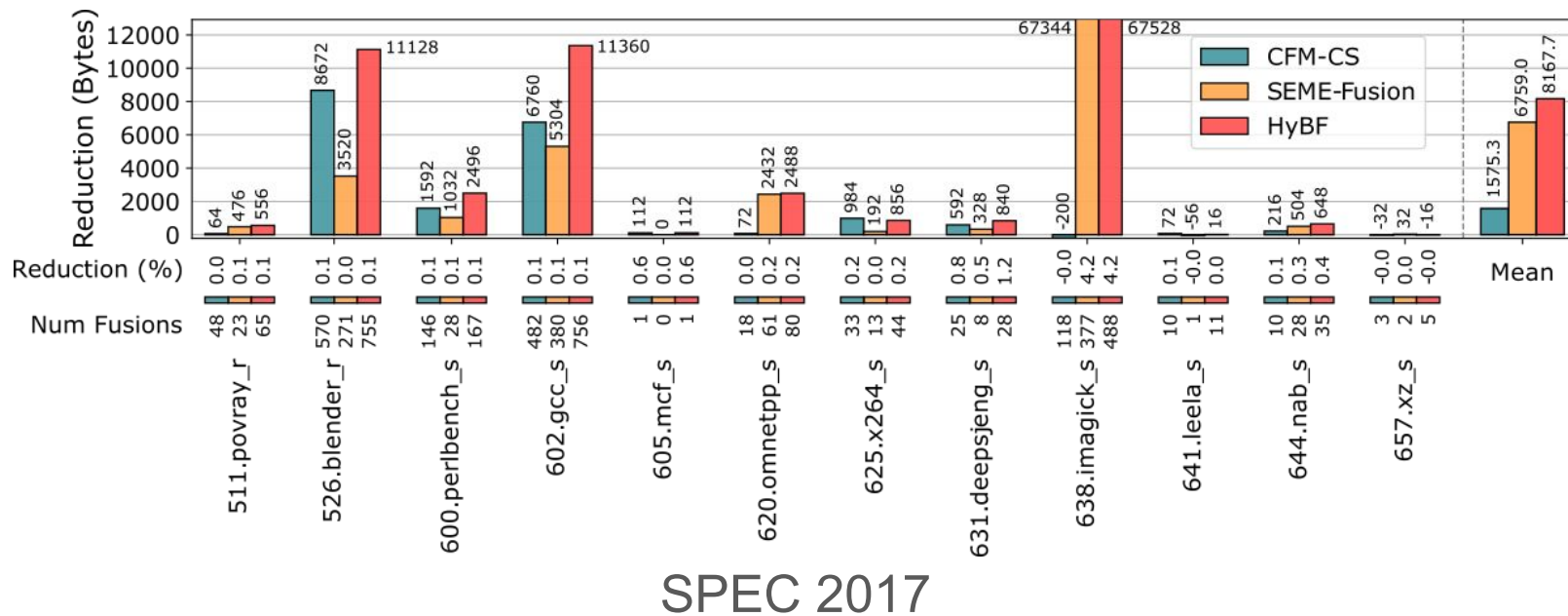
4% reduction in blowfish.

Mibench



Size Reduction

HyBF achieves the best reduction.

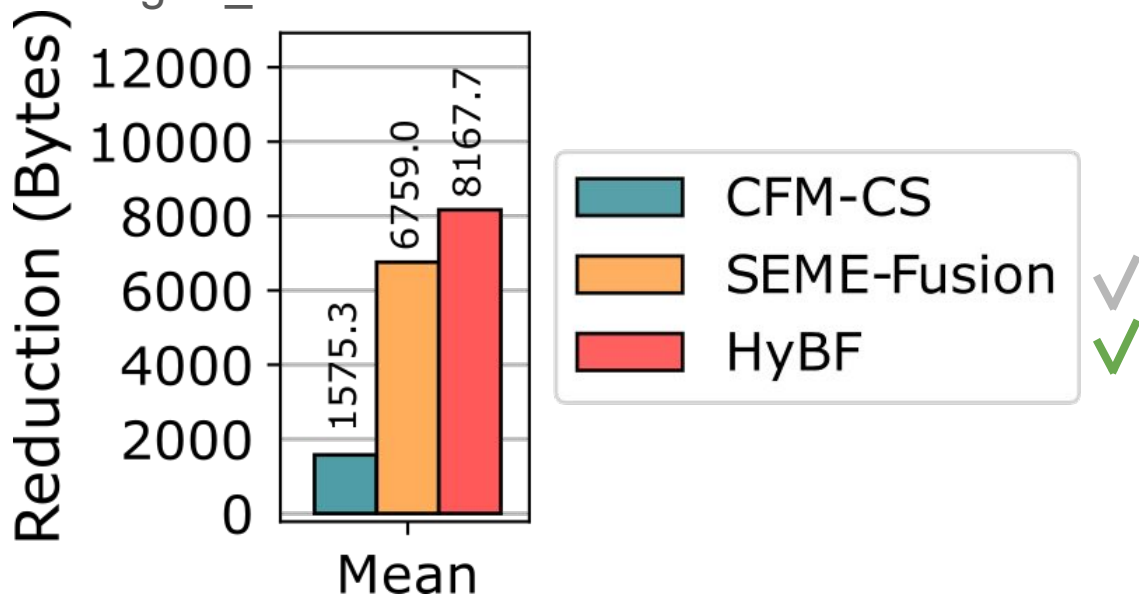


Size Reduction

HyBF achieves the best reduction.

4.2% reduction in 638.imagick_s.

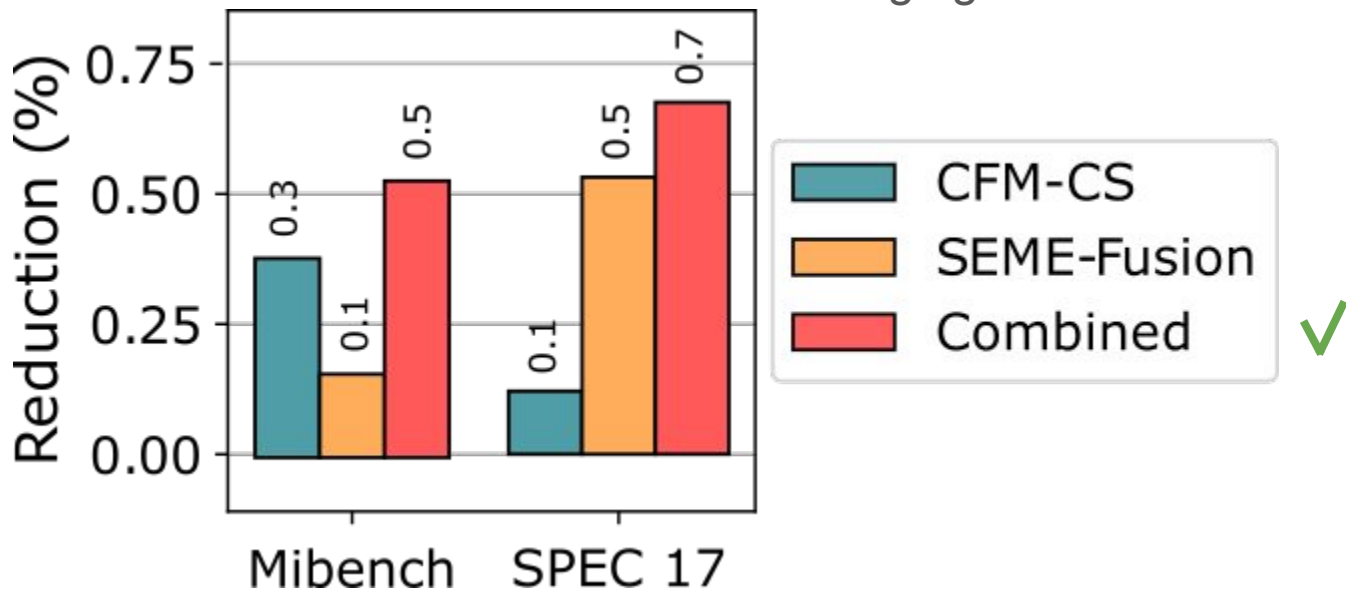
SPEC 2017



Size Reduction

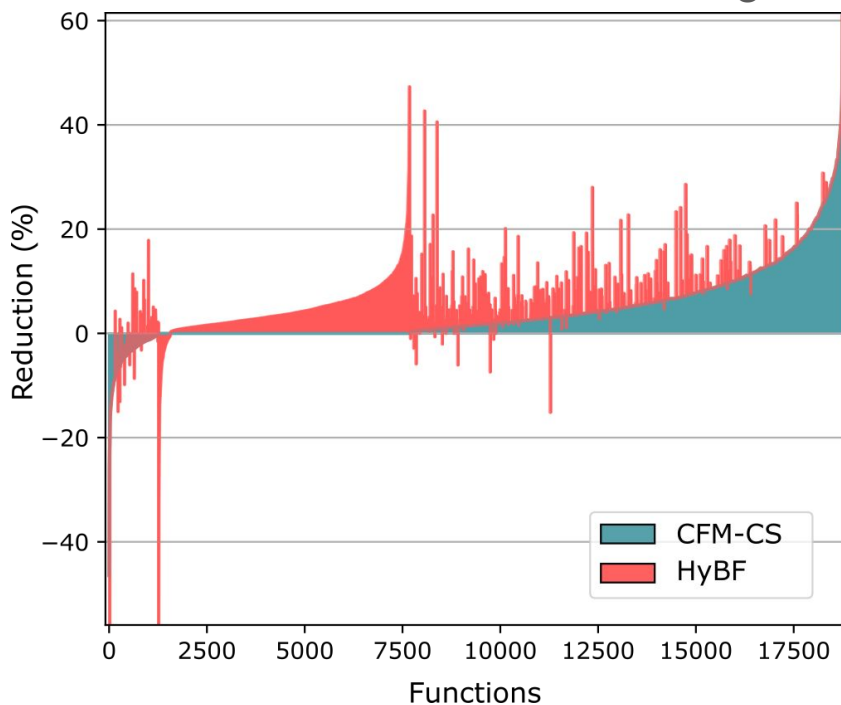
Branch Fusion is complementary to Function Merging:

All versions include the state-of-the-art function merging in LTO mode.



Size Reduction

HyBF tends to give better results than CFM-CS in AnghaBench functions.





HyBF: A Hybrid Branch Fusion Strategy for Code Size Reduction

R. Rocha, C. Saumya, K. Sundararajah, P. Petoumenos, M. Kulkarni, M. O'Boyle

<https://github.com/charitha22/hybf-cc23-artifact>