Differential Outlining

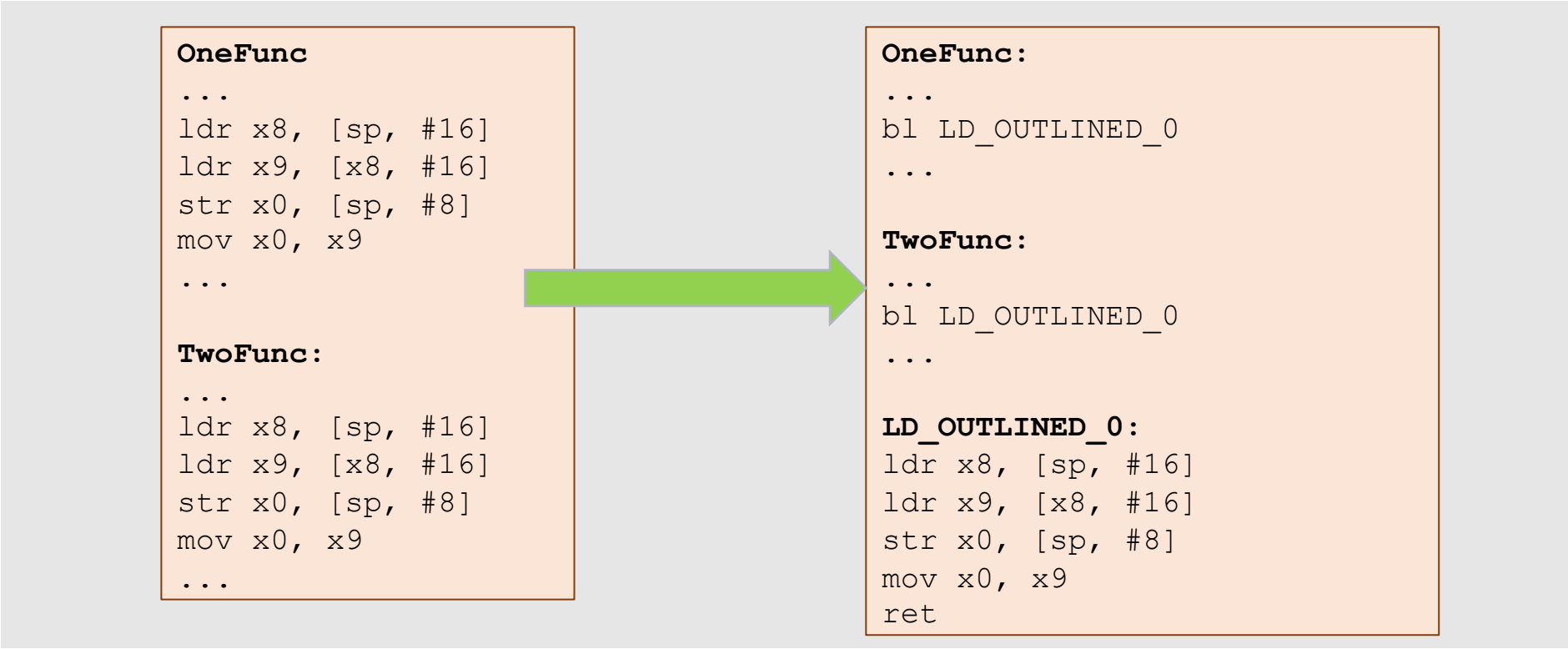
Outlining similar instruction sequences

Girish Mururu, Gai Liu, Chengyan Zhao
Bytedance (Tiktok)
USA

ByteDance

Outlining

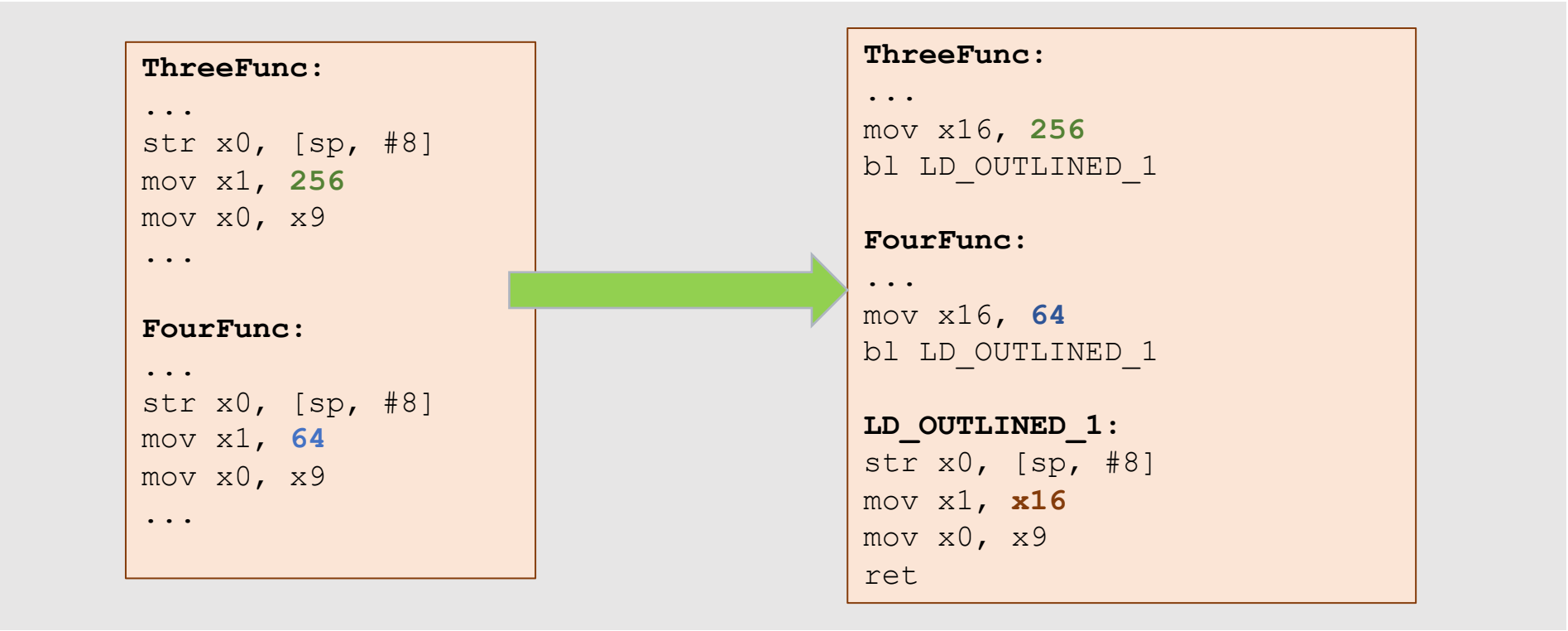
- Find repeated instruction sequence
- Outline the sequence
- Replace the original sequence with a jump(call) to the outlined sequence.



ByteDance

Differential Outlining

- Look for similar sequence rather than the same sequence of instructions
- Outline sequence with generics (replacing differences)
- Replace the original sequence with a jump to the outline and accounting the differences
- Various differences numerous outlining possibilities



ByteDance

Outlining vs Differential Outlining

1

Same seq vs Similar seq

Same sequence is easy to define and group

2

Different kinds of Similarity

Sequences can differ in different ways giving rise to different similar sequences.

3

Hashing

A normal hash does not work for similar sequences. Harder to avoid pairwise comparisons.

4

Outlining

Other than instructions to jump to the outline, Differential Outlining must also account for differences.

Kinds of Differential

1. **Instruction Differential**

   The sequences differ at instruction level, in other words, one of the sequences is a subset of the other.

2. **Register/Immediate Differential**

   The sequences only differ by the registers and immediates that are used in the sequences.

3. **Instruction and Register Differential**

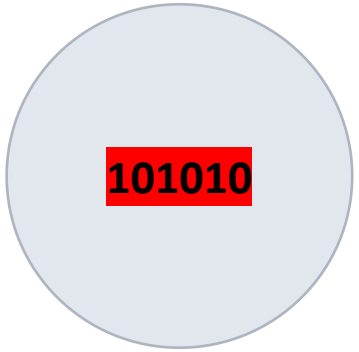
   The sequences differ by both instructions and registers/immediates, however, in terms of opcodes, one sequence is a subset of the other.

4. **Unsubsumed Differential**

   The instructions are different in each sequence such that one sequence is not strictly subsumed by the other in instructions.

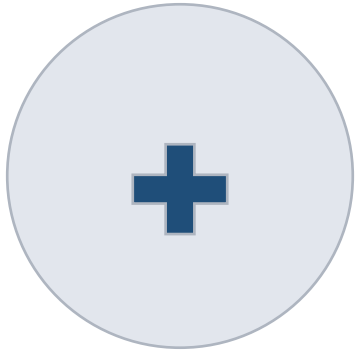
Finding similar sequences

Comparing instructions between sequences is not scalable. The sequences can match at different positions.



Similarity Preserving Encoding

We need hash of the sequences such that similar sequences generate close hashes, that is , the hamming distance between the similar sequences is small or close to 0.



Clustering for scale

Similar sequences must be clustered so that each group can be outlined.

Brute force comparisons of all sequence encodings do not scale either.

Candidate comparisons

Compare the sequences in the group to determine the outline and calls to the outline.

ByteDance

Similarity Preserving Encoding

- Inspired from Similarity Preserving Hashes (e.g. MSRH-v2)
- Bloom filters to generate a bit encoding
  - Multiple bloom filter hash {bf1, bf2.., bfs}
  - Capture different features of the sequences
- The number of overlapping bits between the encodings - similarity metric
- Constraints for outlining:
  - The order of instructions must be preserved
  - Instructions with same opcodes are deemed similar

```
Func A:
mov x1, x2
mov x10, x2
str x8, [sp]
add x9, x8, #1
adrp x8, 0x4008000
ldrsw x8, [x8, #348]
b 0x6505bc4
```

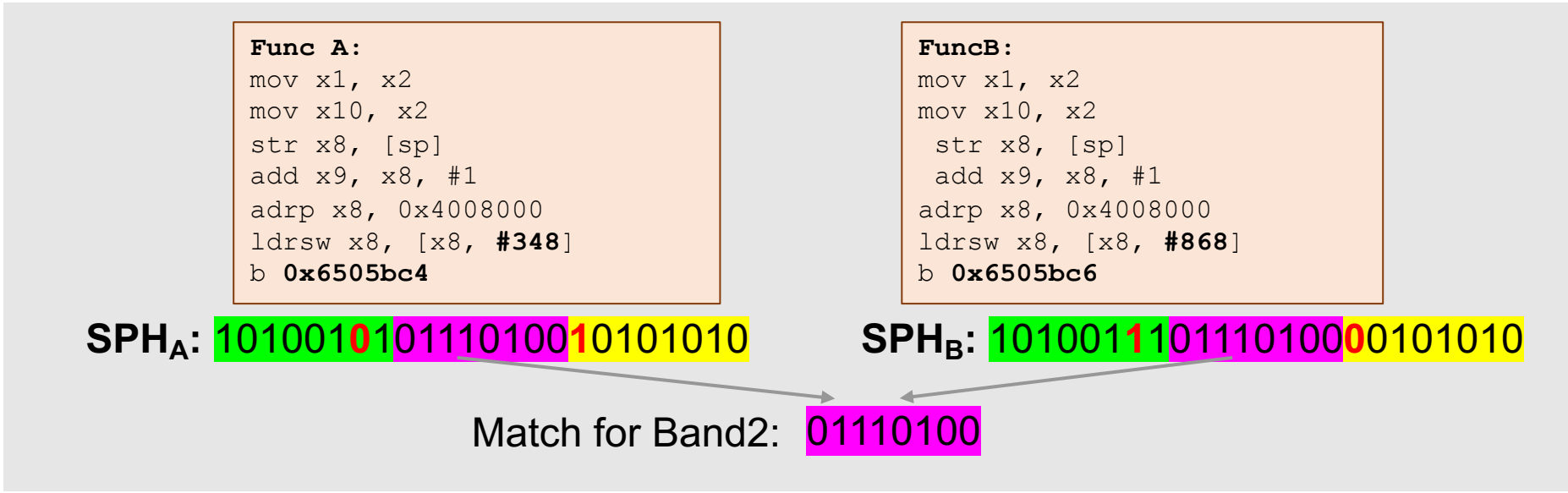
```
FuncB:
mov x1, x2
mov x10, x2
 str x8, [sp]
 add x9, x8, #1
adrp x8, 0x4008000
ldrsw x8, [x8, #868]
b 0x6505bc6
```

SPH_A: 10100101011101001**0**101010 **SPH_B:** 10100111011101000**0**101010

ByteDance

Clustering

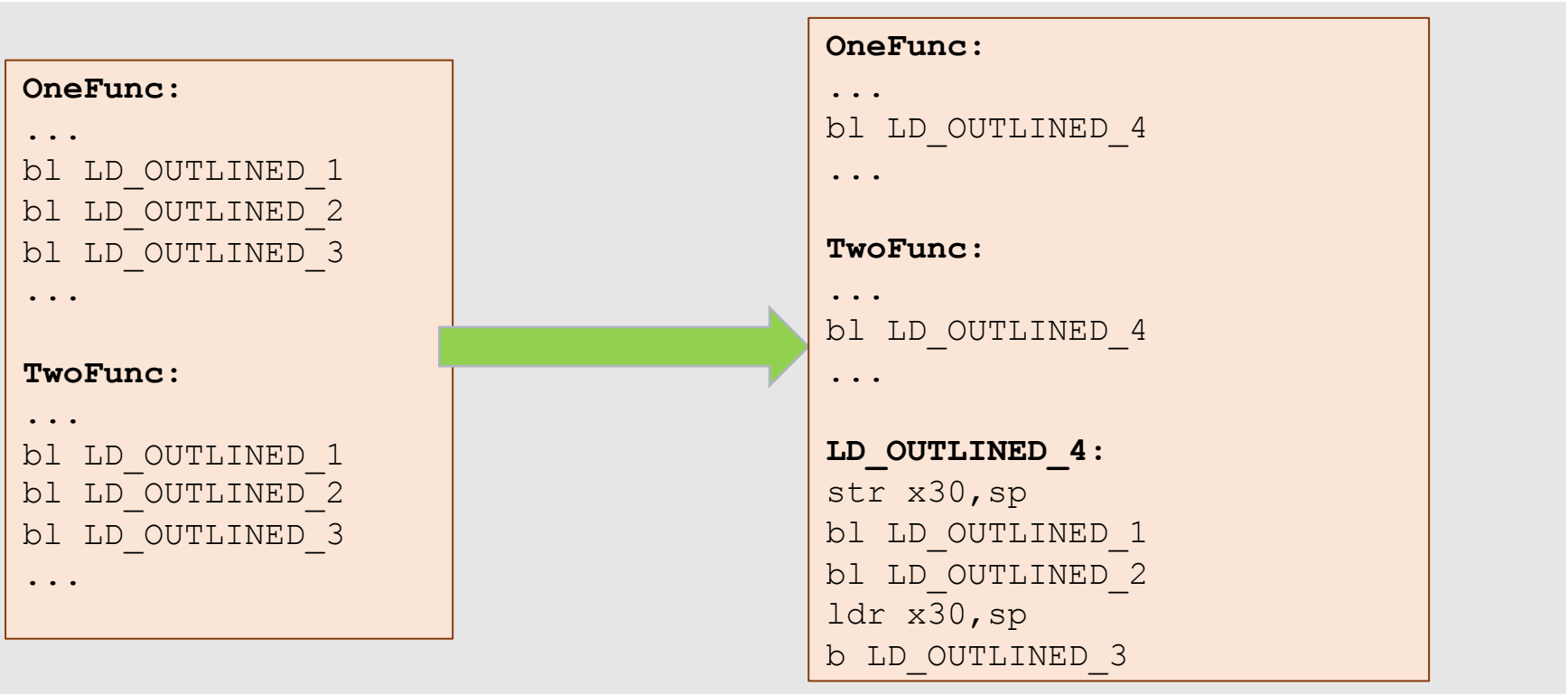
- Group similar sequences
- Overlapping bits in Bloom filters determine similarity
- Locality Sensitive Hashing
  - Hashing for collision (opposite of normal hashing)
  - Encoding already done
  - Banding using chunk of bits as hash functions
  - Candidate for a group: any match in the hash functions



ByteDance

Consecutive Outlined Outlining

- **Instruction Differential:** The sequences differ at instruction level, in other words, one of the sequences is a subset of the other.
- With same sequence outlining, it transforms to consecutive outlined calls
- Outline consecutive outlined calls achieves easier differential outlining
- 2~3% code size savings on social media app.



ByteDance

Similar Function Merging

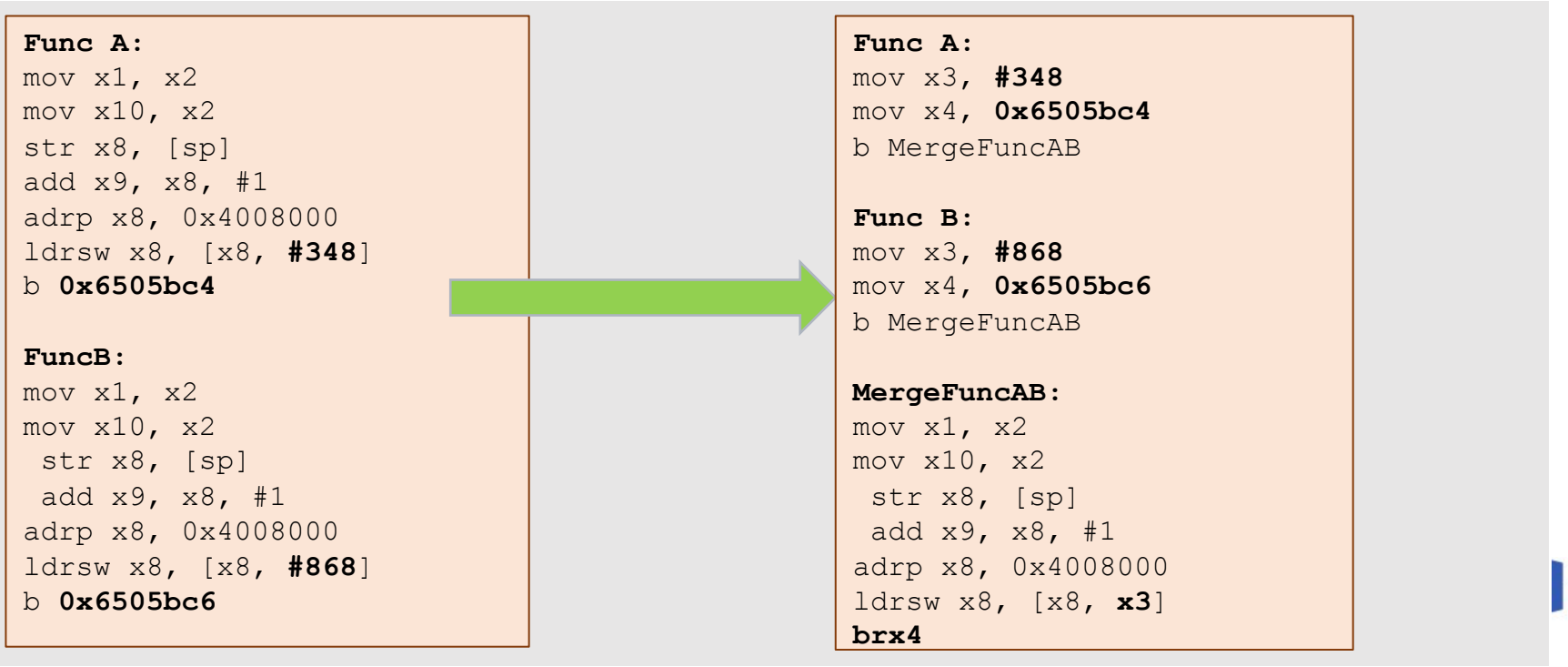
•**Register/Immediate Differential**:The sequences only differ by the registers and immediates that are used in the sequences.

•At function level, the problem transforms to finding similar functions

•Using Similarity Preserving Hashing and Clustering

- Reduces overhead by an order of magnitude in finding similar functions.

•Merging functions that differ by constants and target jump addresses



ByteDance

Similar Sequence Outlining

- **Register/Immediate Differential:** The sequences only differ by the registers and immediates that are used in the sequences.
- At sequence level, the problem transforms to finding similar sequences.
  - Sequences need not be of same length.
  - Found by Similarity preserving hashing.
- Using Similarity Preserving Hashing and Clustering, reduce overhead by an order of magnitude in finding similar sequences.
- Finding similar sequences technique discovers different similar sequences which can then be outlined.
- Future work after similar function merging

Thank You

The **takeaway** is the **new way of finding similar sequences** using **encoding** and **clustering** that can discover new opportunities for code size reduction and elsewhere.

ByteDance