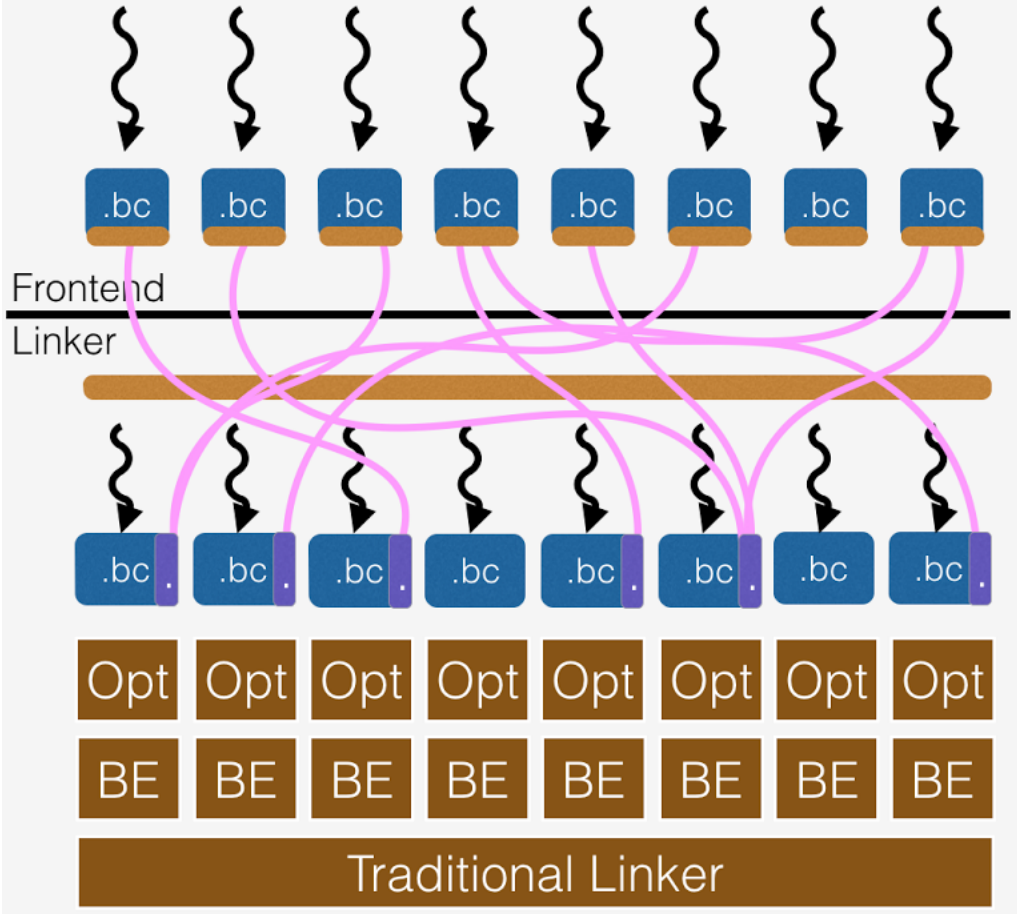
Porting “Merge Similar Functions” pass to ThinLTO

Aditya Kumar

Facebook

ThinLTO (Intro)



Fully-parallel frontend processing
+ initial optimizations

Extra per-function summary informations
are generated *"on the side"*

Link only the summary info
in a giant index: *thin-link*

No need to parse the bitcode

Fully-parallel cross-module function
import based on summary

Imported functions are dropped after inlining

Fully-parallel (very boring) usual
optimizations and code generation

Source: http://blog.llvm.org/2016/06/thinlto-scalable-and-incremental-lto.html

Function Merging (Intro)

- Inter-procedural pass:
  - Merge Identical Functions (llvm/lib/Transforms/IPO/MergeFunctions.cpp)
  - **Merge Similar Functions**
    - https://reviews.llvm.org/D22051
    - https://llvm.org/devmtg/2013-11/slides/Koch-FunctionMerging.pdf
- Deduplicates common parts of similar functions to a separate function
- Code size optimization
- Specially useful for template heavy C++ code bases

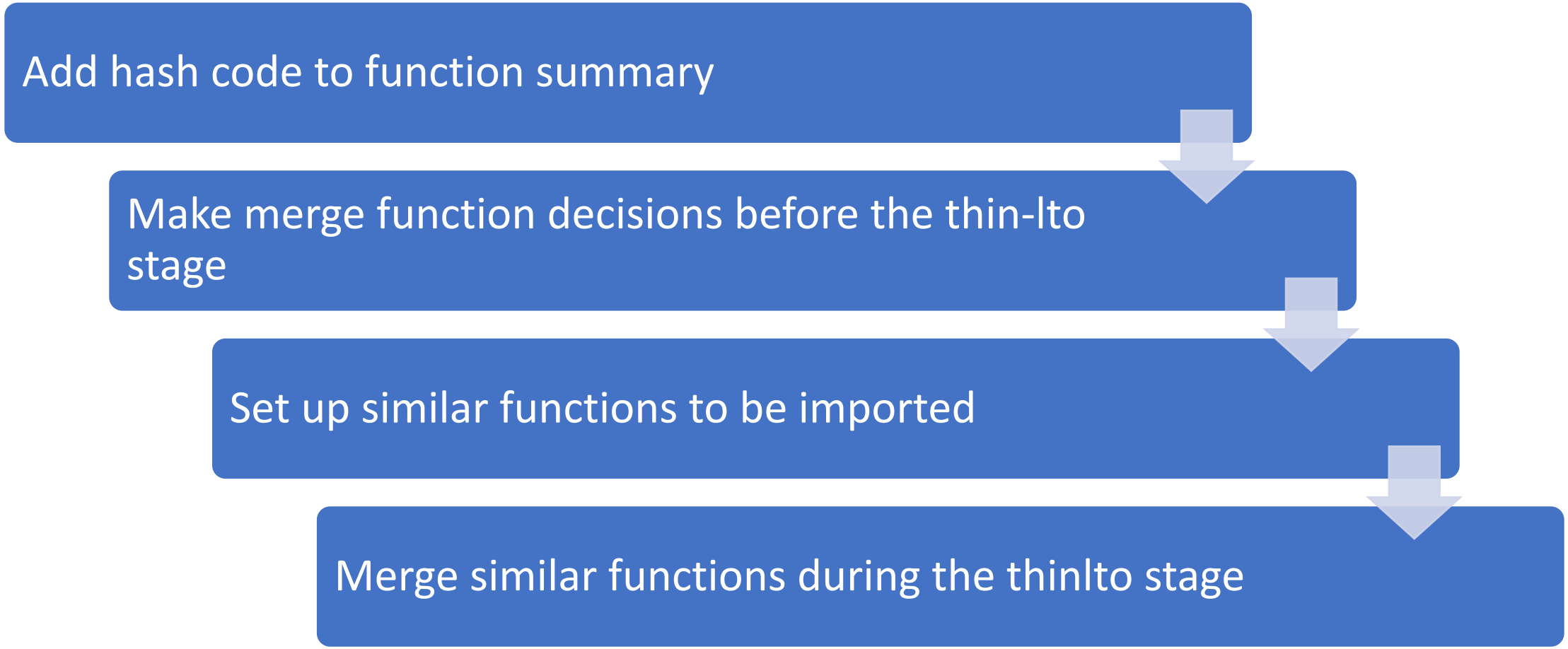
Example

	Before	After

a.c	`int a_foo(int *a, int N) {`
	`  int sum = 0;`
	`  for (int i = 0; i < N; ++i)`
	`  sum += a[i];`
	`  return sum;`
	`}`
		`int a_foo__merged(int *a, int N) {`
		`  int sum = 0;`
		`  for (int i = 0; i < N; ++i)`
		`  sum += a[i];`
		`  return sum;`
		`}`
		
		`int a_foo(int *a, int N) {`
		`  return a_foo__merged(a, N);`
		`}`

b.c	`int b_foo(int *a, int N) {`
	`  int sum = 0;`
	`  for (int i = 0; i < N; ++i)`
	`  sum += a[i];`
	`  return sum;`
	`}`
		`int b_foo(int *a, int N) {`
		`  return a_foo__merged(a, N);`
		`}`

Steps to port Merge Similar Functions to ThinLTO



Add hash code to function summary

- Hash some structural and some semantic properties of a function
  - Number of Basic Blocks, calling convention, Has Var-args etc.
  - Type id of all formal parameters, and return type
- Hashing is fragile to match similar functions

Add hash code to function summary

- Modifying Bitcode reader+writer
- BitcodeReader.cpp
  - Read **SimHash** when bitcode is: FS_PERMODULE or FS_COMBINED
  - **ProcessThinLTOModule**
- BitcodeWriter.cpp
  - **writePerModuleFunctionSummaryRecord**

Make merge function decisions

- Populating Module Summary Index
  - std::map <unsigned, std::vector<GlobalValue::GUID>> SimilarFunctions;
  - std::set <GlobalValue::GUID> HostSimilarFunction;
- Decide which module is the host
- Call **computeMergeSimilarFunctions** before the parallel thinlto stage

Set up similar functions to be imported

- FunctionImport.cpp
  - Inter-procedural, sequential pass pre-thinlto
- For each hosting function in a module, set up similar functions to be imported.
  - Get hosting functions for a module
  - Find all the functions similar to the hosting function from Module Summary Index
  - Import the function (and necessary declarations)

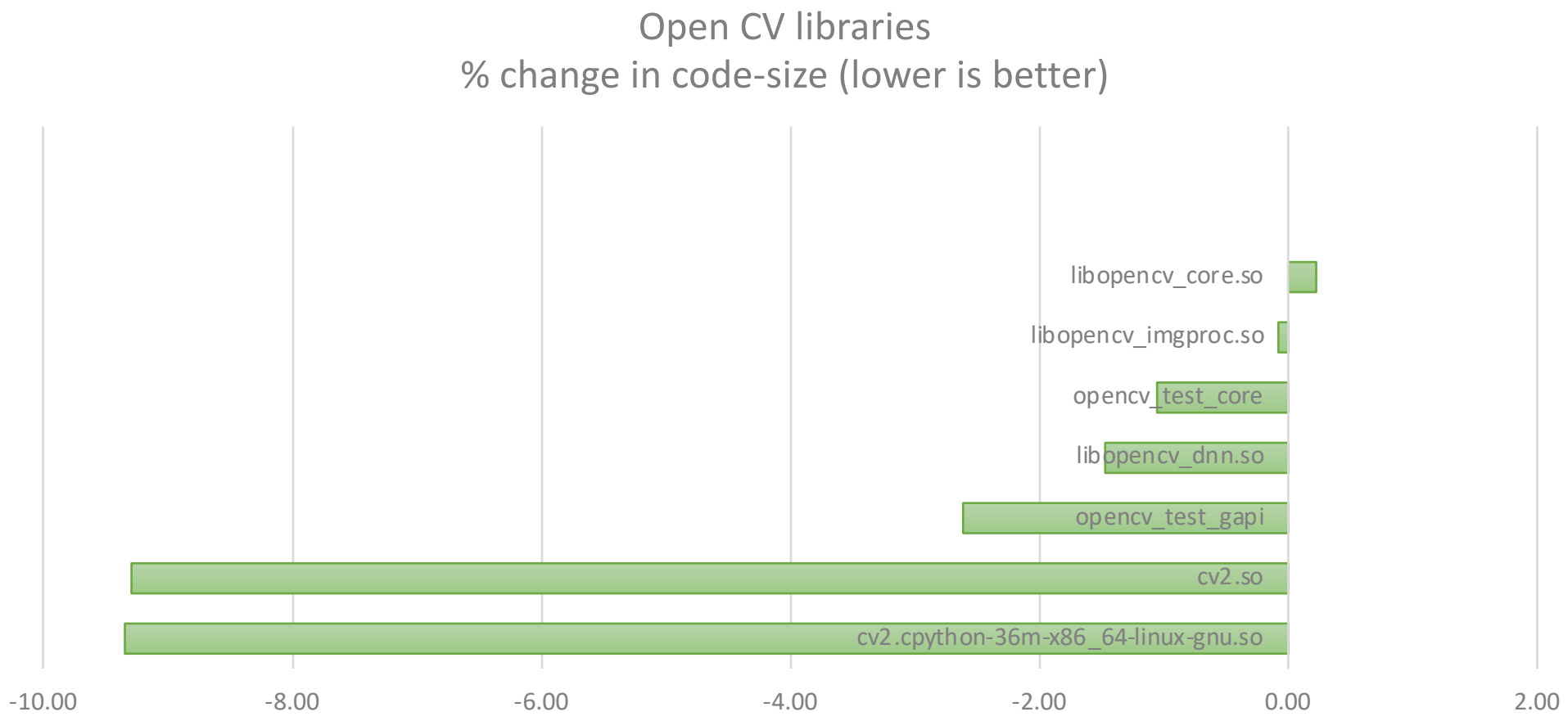
Merge similar functions during the thinlto

- Schedule the pass in thinlto pipeline
  - After Inliner
- Codegen passes run in parallel

Using thinlto in open source projects

- Gold plugin does not take ‘-Os, -Oz’ flags. So ‘-O3’ was used for testing.
- Hard to find projects with ‘simple’ build rules
- cmake passes linker flags to ‘ar’ e.g., -flto=thin
- Need to use llvm-ar and llvm-ranlib

Experimental Results (Open CV Libraries)



Tuning (Cost Models)

- Disable (forced) inlining for widely used function templates
  - C++ Standard library functions like: **find**, **all_of**, **any_of**
- Profile Guided hosting of outlined function to reduce page faults
- Minimum size of functions that should be merged
- Dis-similarity

Patches in Review

- https://reviews.llvm.org/D52896
- https://reviews.llvm.org/D52898
- https://reviews.llvm.org/D52966
- https://reviews.llvm.org/D53253
- https://reviews.llvm.org/D53254

Facebook is Hiring…