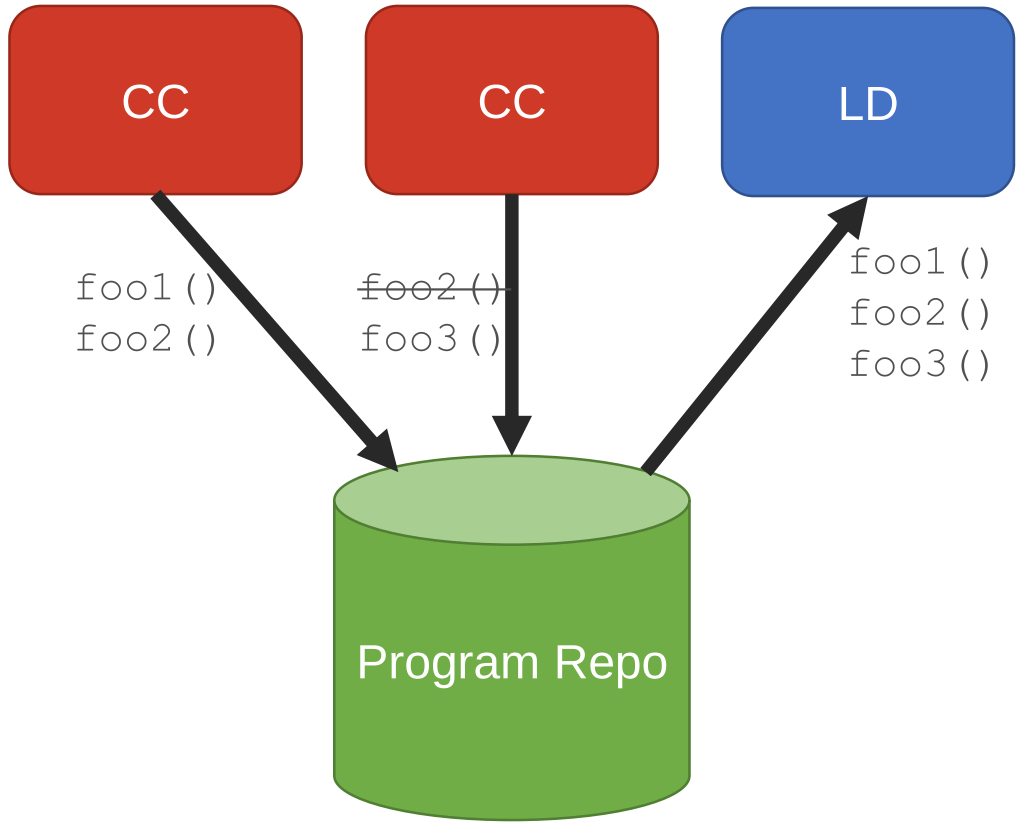
Targeting a statically compiled program repository with LLVM

Russell Gallop

April 2019

Program Repository

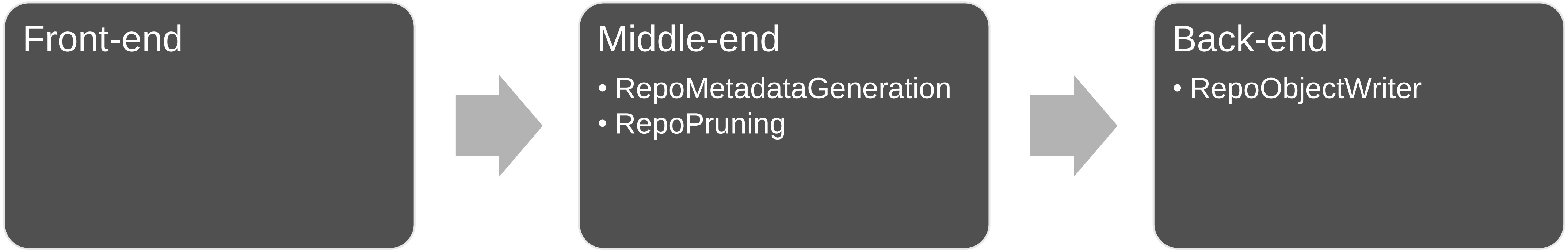
- The **Program Repository (or Repo)** is a research project at SN Systems
- It aims to dramatically improve build times for large C++ programs by:
  - Avoiding repeated codegen across compilation units and builds
  - Moving link time de-duplication to compile time
- Stores compiled objects in a repository instead of object files



History

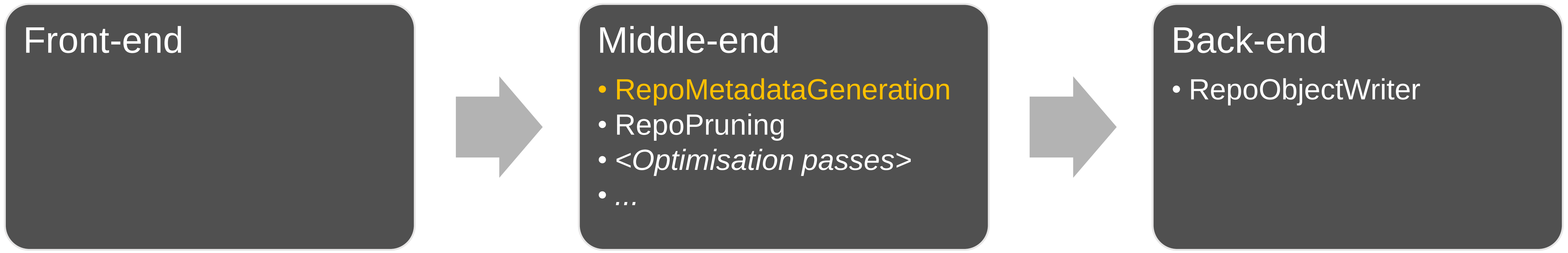
- At the 2016 US Dev Meeting talk we demoed “Toy tools” prototype
  - https://www.youtube.com/watch?v=-pL94rqyQ6c
  - This used a toy programming language and YAML object files
- Since then we have implemented this idea for C/C++ and Linux on a fork of LLVM:
  - https://github.com/SNSystems/llvm-project-prepo
  - Up to date with 8.0 release branch point

Implementation



- We implemented this as a couple of optimization passes and a new object type

a) Adding Program Repository metadata



- We added a new pass to the start of the optimisation pipeline:

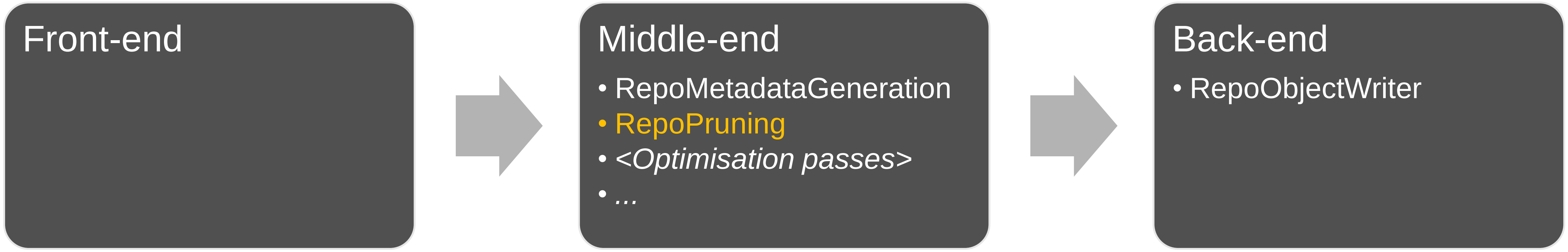
```
    class RepoMetadataGeneration : public ModulePass {...}
```

- This calculates a digest of each function from the front-end and the pass pipeline that will be run on it
- Recorded as metadata in the IR

```
    !2 = !TicketNode(name: "_Z3foov",
           digest: [16 x i8] c"0g4WG\1B&\89\F9\FB\92|\AA\94j\9B",
           linkage: external,
           pruned: false)
```

- This digest is used as the key for the compiled object data in the Program Repo

b) Pruning



- Following this we added another pass

```
     class RepoPruning : public ModulePass {...}
```

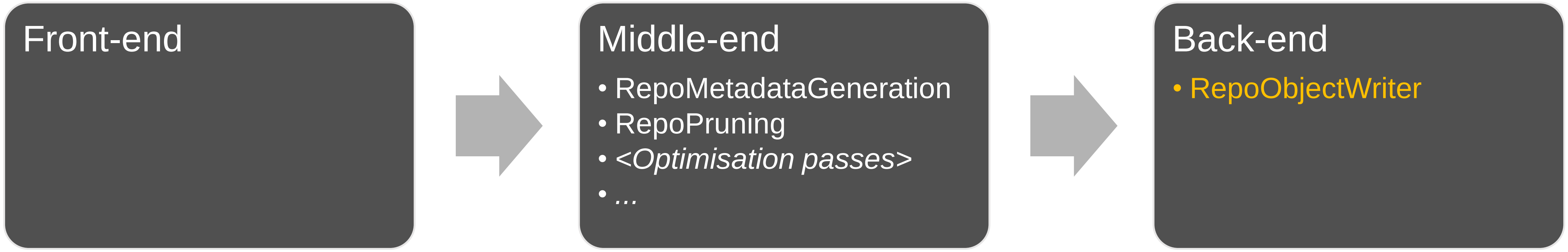
- This checks if compiled objects are already in the Program Repo
- If present then it avoids optimisation by:
  - Setting their linkage type

```
  define available_externally void @_Z3foov() #0 !repo_ticket !2
```

  - Marking that they have been pruned in the Program Repo metadata

```
  !2 = !TicketNode(name: "_Z3foov",
         digest: [16 x i8] c"0g4WG\1B&\89\F9\FB\92|\AA\94j\9B",
         linkage: external,
         pruned: true)
```

c) Emitting objects to the repository



- We have added a new ObjectWriter
  - `class` **`RepoObjectWriter`** `: public MCObjectWriter {`
- This writes 2 things to the Program Repo
  1. Compiled objects (called `Fragment`s) indexed by the object digest
  2. A list of all compiled objects in a module (a `CompilationRecord`)
- In place of an object file it writes a small output file (a `TicketFile`)
  - This has a file signature and the index of the module's `CompilationRecord` (e.g.)

```
$ xxd foo.o
    00000000: 746b 6354 6f70 6552 15ae 9e73 ff59 92ee  tkcTopeR...s.Y..
    00000010: 874c 2a27 e9a0 bf50                      .L*'...P
```

What about linking?

- Program Repo fundamentally breaks the traditional object file format so requires a different approach to linking
- We have started work on a prototype linker to link programs directly from the Program Repo

Testing

- For testing we have a tool called `repo2obj`. This:
  - Reads a TicketFile
  - Finds all the objects that are required for it in the Program Repo
  - Creates ELF object files which can be linked with a standard ELF linker
- This is inefficient as it creates all of the duplicates that the repository tries to avoid but allows us to test the compiler and repository are working correctly

Results

- We can now build optimized LLVM/Clang with the Program Repo
  - ~100 LIT/unit test failures, being investigated
- Limited debug information (line tables)
- Working on performance results

Summary

- Program Repository concept implemented in LLVM for Linux and C/C++
- Added 2 ModulePasses and one ObjectWriter
- We can build and run optimized LLVM/Clang (with `repo2obj`)
- Please try it out: https://github.com/SNSystems/llvm-project-prepo

- Thanks to:
  - Paul Bowen-Huggett
  - Phil Camp
  - Maggie Yi
  - Carlos Enciso