ART OptimizingCompiler 简析

中科院软件所PLCT实验室 史宁宁

目录

自我介绍

自我介绍

- 知乎：小乖他爹
- BILIBILI: 小乖他爹-知乎
- CSDN：snsn1984
- Github：shining1984
- 个人talk库：https://github.com/shining1984/talks

自我介绍


华为方舟编译器之美
基于开源代码的架构分析与实现
史宁宁◎编著
清华大学出版社
联袂推荐
THE BEAUTY OF HUAWEI OPENARKCOMPILER
THE ANALYSIS AND IMPLEMENT OF ARCHITECTURE BASED OPEN SOURCE CODE
华为方舟编译器之美
基于开源代码的架构分析与实现
史宁宁◎编著
清华大学出版社

OptimizingCompiler基础

OptimizingCompiler概念

- OptimizingCompiler可以从宏观和微观两方面理解;
- 宏观上，OptimizingCompiler可以被视为一个完整的编译器，将输入dex格式文件编译为oat格式，是dex2oat工具的核心;
- 微观上， OptimizingCompiler是一个具体的类，是Compiler的子类。

宏观OptimizingCompiler

- 宏观OptimizingCompiler负责dex2oat中从dex到oat的编译过程；
- 宏观OptimizingCompiler与dex2oat在不同的目录，宏观的OptimizingCompiler位于art/compiler/optimizing目录，dex2oat位于art/dex2oat目录；
- 宏观的OptimizingCompiler可以看做是OptimizingCompiler类及其相关的类所构成的一个编译器。

微观OptimizingCompiler

- 微观OptimizingCompiler是一个类;
- OptimizingCompiler类是ART中Compiler类的唯一一个子类;

class OptimizingCompiler final : public Compiler {

Compiler::Create方法

- Compiler* Compiler::Create(const CompilerOptions& compiler_options, CompiledMethodStorage* storage, Compiler::Kind kind) {
- ...
- switch (kind) {
- case kQuick:
- // TODO: Remove Quick in options.
- case kOptimizing:
- return CreateOptimizingCompiler(compiler_options, storage);

- default:
- LOG(FATAL) << "UNREACHABLE";
- UNREACHABLE();
- }
- }

OptimizingCompiler实现

正常编译

art/compiler/optimizing/optimizing_compiler.cc

```
CompiledMethod* OptimizingCompiler::Compile(
                                        const dex::CodeItem* code_item,
                                         uint32_t access_flags,
                                         InvokeType invoke_type,
                                         uint16_t class_def_idx,
                                         uint32_t method_idx,
                                         Handle<mirror::ClassLoader> jclass_loader,
                                         const DexFile& dex_file,
                                         Handle<mirror::DexCache> dex_cache) const;
```

Jni编译

art/compiler/optimizing/optimizing_compiler.cc

```
CompiledMethod* OptimizingCompiler::JniCompile(
        uint32_t access_flags,
        uint32_t method_idx,
        const DexFile& dex_file,
        Handle<mirror::DexCache> dex_cache) const;
```

正常编译-SSA树构建

art/compiler/optimizing/builder.cc

GraphAnalysisResult HGraphBuilder::BuildGraph() ;

正常编译-公共优化环节

art/compiler/optimizing/optimizing_compiler.cc

```
void OptimizingCompiler::RunOptimizations(HGraph* graph,
                                          CodeGenerator* codegen,
                                          const DexCompilationUnit& dex_compilation_unit,
                                          PassObserver* pass_observer,
                                          VariableSizedHandleScope* handles) const;
```

正常编译-目标平台优化环节

art/compiler/optimizing/optimizing_compiler.cc

```
bool OptimizingCompiler::RunArchOptimizations(HGraph* graph,
                                              CodeGenerator* codegen,
                                              const DexCompilationUnit& dex_compilation_unit,
                                              PassObserver* pass_observer,
                                              VariableSizedHandleScope* handles) const;
```

正常编译-寄存器分配

art/compiler/optimizing/optimizing_compiler.cc

```
static void AllocateRegisters(HGraph* graph,
                              CodeGenerator* codegen,
                              PassObserver* pass_observer,
                              RegisterAllocator::Strategy strategy,
                              OptimizingCompilerStats* stats);
```

正常编译-代码生成

art/compiler/optimizing/code_generator.cc

void CodeGenerator::Compile(CodeAllocator* allocator);

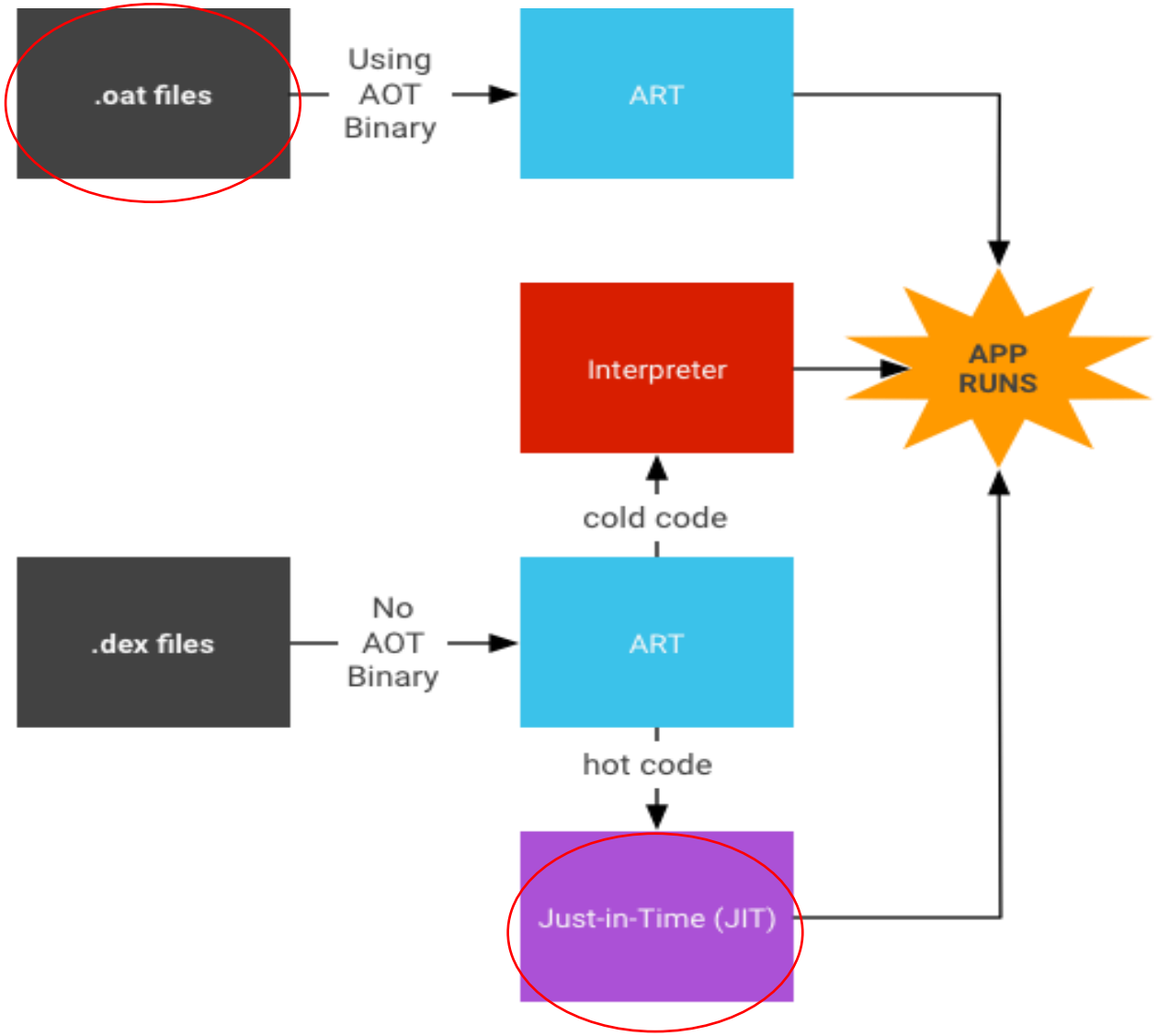
正常编译- Emit环节

art/compiler/optimizing/optimizing_compiler.cc

```
CompiledMethod* OptimizingCompiler::Emit(
                    ArenaAllocator* allocator,
                    CodeVectorAllocator* code_allocator,
                    CodeGenerator* codegen,
                    const dex::CodeItem* code_item_for_osr_check) const;
```

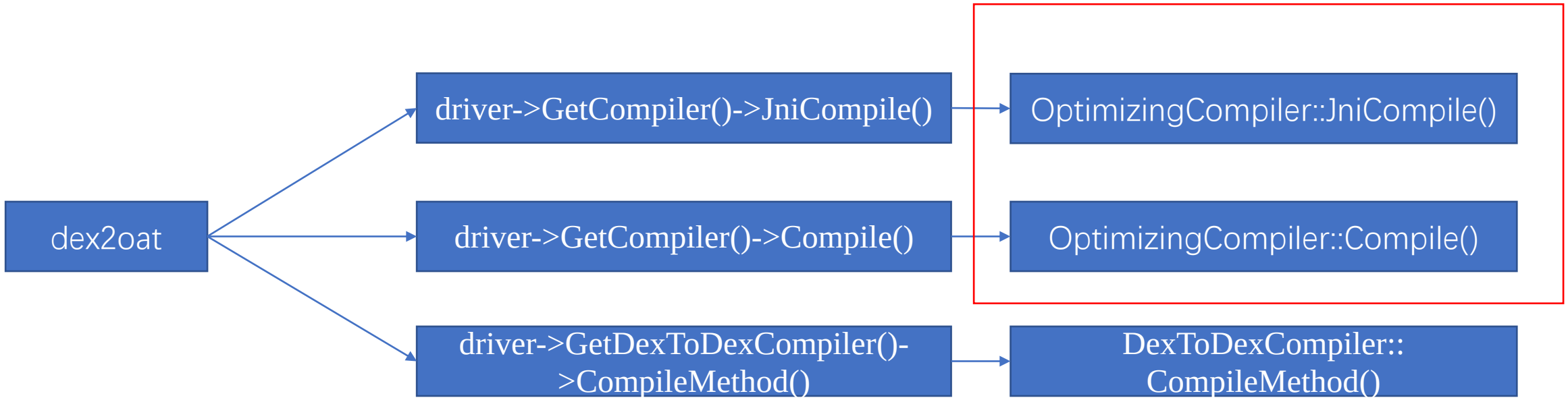
OptimizingCompiler应用

OptimizingCompiler应用

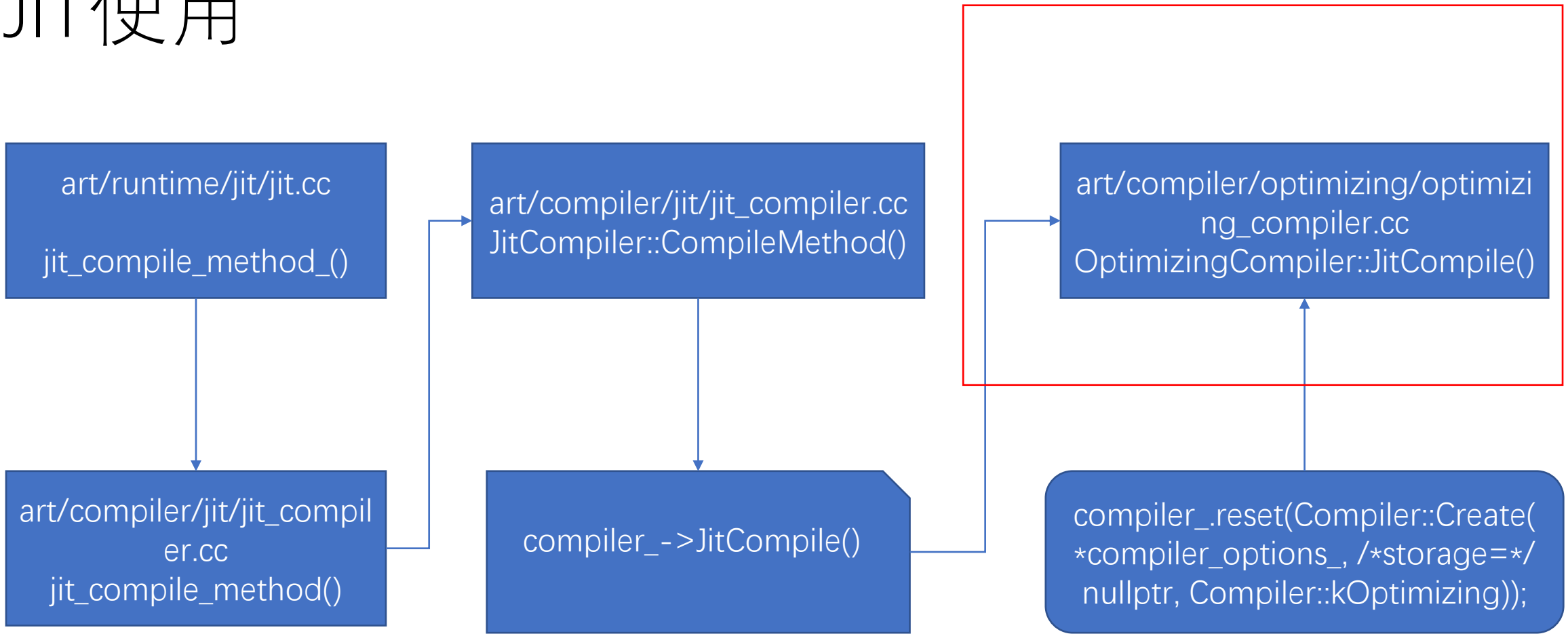


From:
https://source.android.com/devices/tech/dalvik/jit-compiler

dex2oat


dex2oat
driver->GetCompiler()->JniCompile()
driver->GetCompiler()->Compile()
driver->GetDexToDexCompiler()->CompileMethod()
OptimizingCompiler::JniCompile()
OptimizingCompiler::Compile()
DexToDexCompiler::CompileMethod()

JIT使用


art/runtime/jit/jit.cc
jit_compile_method_()
art/compiler/jit/jit_compiler.cc
jit_compile_method()
art/compiler/jit/jit_compiler.cc
JitCompiler::CompileMethod()
compiler_->JitCompile()
art/compiler/optimizing/optimizing_compiler.cc
OptimizingCompiler::JitCompile()
compiler_.reset(Compiler::Create(*compiler_options_, /*storage=*/ nullptr, Compiler::kOptimizing));

Thanks!