

Building an LLVM-based tool

Lessons Learned

EuroLLVM 2019, Brussels

whoami

- Alex Denisov
- Software Engineer at PTScientists GmbH
- LLVM Hacker
- <https://lowlevelbits.org>
- https://twitter.com/1101_debian

Agenda

- Build System
- Memory Management
- Parallelization
- Multi Version Support
- Multi OS Support
- Bitcode Extraction
- And more

Mull

- <https://github.com/mull-project/mull>
- Mutation Testing: Leaving the Stone Age. FOSDEM 2017
<https://www.youtube.com/watch?v=YEgiiylCkpQ>
- Mull it over: mutation testing based on LLVM
<https://ieeexplore.ieee.org/document/8411727>
- Works on Linux, macOS, FreeBSD
- Works with LLVM 3.9 - 8.0



Disclaimer

I am an Expert

=>

Blindly Follow My Advice!

Disclaimer

~~I am an Expert
→
Blindly Follow My Advice!~~

YMMV

An LLVM-based tool

- Works with LLVM Bitcode
- Load
- Analyze
- Transform
- Process and report results

llvm-config

```
> clang -c `llvm-config --cxxflags` \  
foo.cpp -o foo.o
```

```
> clang -c `llvm-config --cxxflags` \  
bar.cpp -o bar.o
```

```
> clang `llvm-config --ldflags` \  
`llvm-config --libs core support` \  
bar.o foo.o -o foobar.bin
```


llvm-config

```
> llvm-config --cxxflags  
-I/opt/llvm/6.0.0/include  
...  
-Werror=unguarded-availability-new  
-O3 -DNDEBUG  
...
```

llvm-config

```
> llvm-config --cxxflags  
-I/opt/llvm/6.0.0/include  
...  
-Werror=unguarded-availability-new  
-O3 -DNDEBUG  
...
```

llvm-config

```
> llvm-config --libs core  
-lLLVMCore  
-lLLVMBinaryFormat  
-lLLVMSupport  
-lLLVMDemangle
```

llvm-config

```
/usr/lib/llvm-4.0/lib/libLLVM.dylib  
/usr/lib/llvm-6.0/lib/libLLVM.dylib
```

llvm-config

```
/usr/lib/llvm-4.0/lib/libLLVM.dylib  
/usr/lib/llvm-6.0/lib/libLLVM.dylib
```

```
> clang foo.o bar.o -lLLVMsupport -o foobar.bin
```

```
> ./foobar.bin
```

```
LLVM ERROR: inconsistency in registered CommandLine  
options
```

llvm-config

```
/usr/lib/llvm-4.0/lib/libLLVM.dylib  
/usr/lib/llvm-6.0/lib/libLLVM.dylib
```

```
> clang foo.o bar.o -lLLVM -o foobar.bin
```

```
> ./foobar.bin
```

```
All good.
```

llvm-config

```
if macOS
LDFLAGS=-lLLVM -lclangEdit
else
LDFLAGS=-Wl,--start-group -lLLVM -lclangEdit -Wl,--end-group
endif

clang foo.o bar.o $LDFLAGS -o foobar.bin
```

CMake

<https://llvm.org/docs/CMakePrimer.html>

```
add_executable(foo
    foo.cpp bar.cpp
)
target_include_directories(foo
    /usr/include /usr/local/include
)
target_link_libraries(foo
    m sqlite3 ncurses
)
```


CMake

```
set (search_paths
    ${PATH_TO_LLVM}
    ${PATH_TO_LLVM}/lib/cmake
    ${PATH_TO_LLVM}/lib/cmake/llvm
    ${PATH_TO_LLVM}/lib/cmake/clang
    ${PATH_TO_LLVM}/share/clang/cmake/
    ${PATH_TO_LLVM}/share/llvm/cmake/
)

find_package(LLVM REQUIRED CONFIG
    PATHS ${search_paths}
    NO_DEFAULT_PATH)
```

CMake

```
find_package(LLVM REQUIRED CONFIG  
             PATHS ${search_paths}  
             NO_DEFAULT_PATH)
```

```
find_package(Clang REQUIRED CONFIG  
            PATHS ${search_paths}  
            NO_DEFAULT_PATH)
```

CMake

```
target_include_directories(mull PUBLIC ${LLVM_INCLUDE_DIRS})  
target_link_libraries(mull LLVMSupport clangTooling)
```

CMake

```
target_include_directories(mull PUBLIC ${LLVM_INCLUDE_DIRS})  
target_link_libraries(mull LLVMSupport clangTooling)
```

LLVM ERROR: inconsistency in registered CommandLine options

CMake

```
target_include_directories(mull PUBLIC ${LLVM_INCLUDE_DIRS})

if (LLVM IN_LIST LLVM_AVAILABLE_LIBS)
    target_link_libraries(mull LLVM clangTooling)
else()
    target_link_libraries(mull LLVMSupport clangTooling)
endif()
```

Multiple LLVM Versions

<http://github.com/klee/klee>

```
#if LLVM_VERSION_CODE >= LLVM_VERSION(4, 0)
#include <llvm/Bitcode/BitcodeReader.h>
#else
#include <llvm/Bitcode/ReaderWriter.h>
#endif
```

```
#if LLVM_VERSION_CODE >= LLVM_VERSION(5, 0)
    assert(ii->getNumOperands() == 3 && "wrong number of arguments");
#else
    assert(ii->getNumOperands() == 2 && "wrong number of arguments");
#endif
```

Multiple LLVM Versions

LLVMCompatibility/

├── 3.9.x

| ├── CMakeLists.txt

| ├── LLVMCompatibility.cpp

| └── LLVMCompatibility.h

├── 4.x.x

| ├── CMakeLists.txt

| ├── LLVMCompatibility.cpp

| └── LLVMCompatibility.h

├── 5.x.x

├── 6.x.x

├── 7.x.x

└── 8.x.x

Multiple LLVM Versions

```
if ( EXISTS LLVMCompatibility/${LLVM_VERSION} )
    add_subdirectory(LLVMCompatibility/${LLVM_VERSION} )
else()
    message(FATAL_ERROR
            "LLVM-${LLVM_VERSION} is not supported")
endif()
```


Multiple LLVM Versions

LLVM 3.9

```
#include <llvm/ExecutionEngine/Orc/CompileUtils.h>
#include <llvm/ExecutionEngine/Orc/ExecutionUtils.h>
#include <llvm/ExecutionEngine/Orc/JITSymbol.h>
#include <llvm/ExecutionEngine/RuntimeDyld.h>

namespace llvm_compat {
using namespace llvm;

typedef RuntimeDyld::SymbolResolver SymbolResolver;
typedef RuntimeDyld::SymbolInfo JITSymbolInfo;
typedef orc::JITSymbol JITSymbol;

object::OwningBinary<object::ObjectFile>
  compileModule(orc::SimpleCompiler &compiler,
               Module &module);

std::unique_ptr<Module>
  parseBitcode(MemoryBufferRef bufferRef,
               LLVMContext &context);
}
```

LLVM 8.0

```
#include <llvm/ExecutionEngine/Orc/CompileUtils.h>
#include <llvm/ExecutionEngine/Orc/ExecutionUtils.h>

#include <llvm/ExecutionEngine/RuntimeDyld.h>

namespace llvm_compat {
using namespace llvm;

typedef LegacyJITSymbolResolver SymbolResolver;
typedef JITSymbol JITSymbolInfo;
typedef JITSymbol JITSymbol;

object::OwningBinary<object::ObjectFile>
  compileModule(orc::SimpleCompiler &compiler,
               Module &module);

std::unique_ptr<Module>
  parseBitcode(MemoryBufferRef bufferRef,
               LLVMContext &context);
}
```

Multiple LLVM Versions

LLVM 3.9

```
#include <llvm/ExecutionEngine/Orc/CompileUtils.h>
#include <llvm/ExecutionEngine/Orc/ExecutionUtils.h>
#include <llvm/ExecutionEngine/Orc/JITSymbol.h>
#include <llvm/ExecutionEngine/RuntimeDyld.h>

namespace llvm_compat {
using namespace llvm;

typedef RuntimeDyld::SymbolResolver SymbolResolver;
typedef RuntimeDyld::SymbolInfo JITSymbolInfo;
typedef orc::JITSymbol JITSymbol;

object::OwningBinary<object::ObjectFile>
  compileModule(orc::SimpleCompiler &compiler,
               Module &module);

std::unique_ptr<Module>
  parseBitcode(MemoryBufferRef bufferRef,
              LLVMContext &context);
}
```

LLVM 8.0

```
#include <llvm/ExecutionEngine/Orc/CompileUtils.h>
#include <llvm/ExecutionEngine/Orc/ExecutionUtils.h>
#include <llvm/ExecutionEngine/RuntimeDyld.h>

namespace llvm_compat {
using namespace llvm;

typedef LegacyJITSymbolResolver SymbolResolver;
typedef JITSymbol JITSymbolInfo;
typedef JITSymbol JITSymbol;

object::OwningBinary<object::ObjectFile>
  compileModule(orc::SimpleCompiler &compiler,
               Module &module);

std::unique_ptr<Module>
  parseBitcode(MemoryBufferRef bufferRef,
              LLVMContext &context);
}
```

Multiple LLVM Versions

MCJIT

ORC JIT

Symbol Resolver

Memory Manager

Dynamic Linker

Etc.

Multiple LLVM Versions

MCJIT

Native JIT

ORC JIT

Symbol Resolver

Memory Manager

Dynamic Linker

Etc.

Multiple LLVM Versions

LLVM 3.8

```
Function *  
CloneFunction(const Function *F,  
              ValueToValueMapTy &VMap,  
              bool ModuleLevelChanges,  
              ClonedCodeInfo *CodeInfo);
```

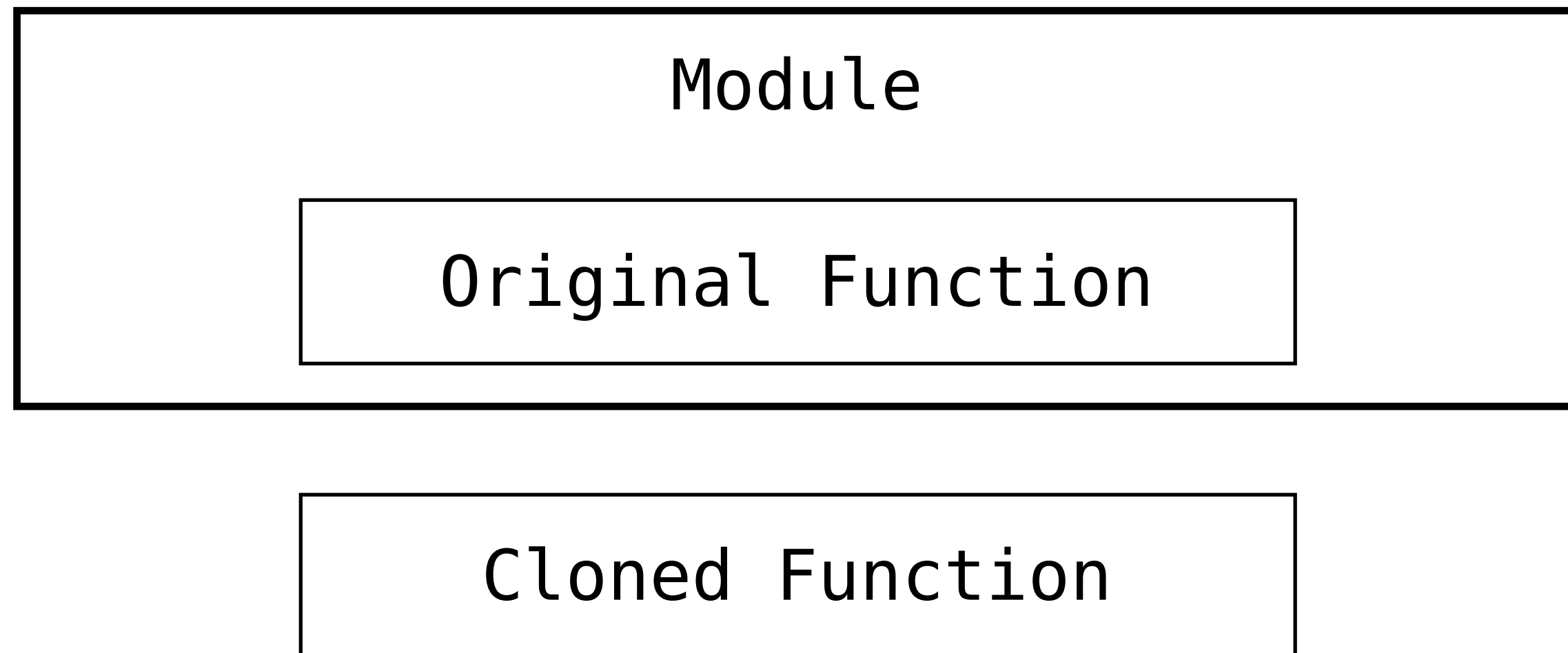
LLVM 3.9+

```
Function *  
CloneFunction(Function *F,  
              ValueToValueMapTy &VMap,  
              ClonedCodeInfo *CodeInfo);
```

Multiple LLVM Versions

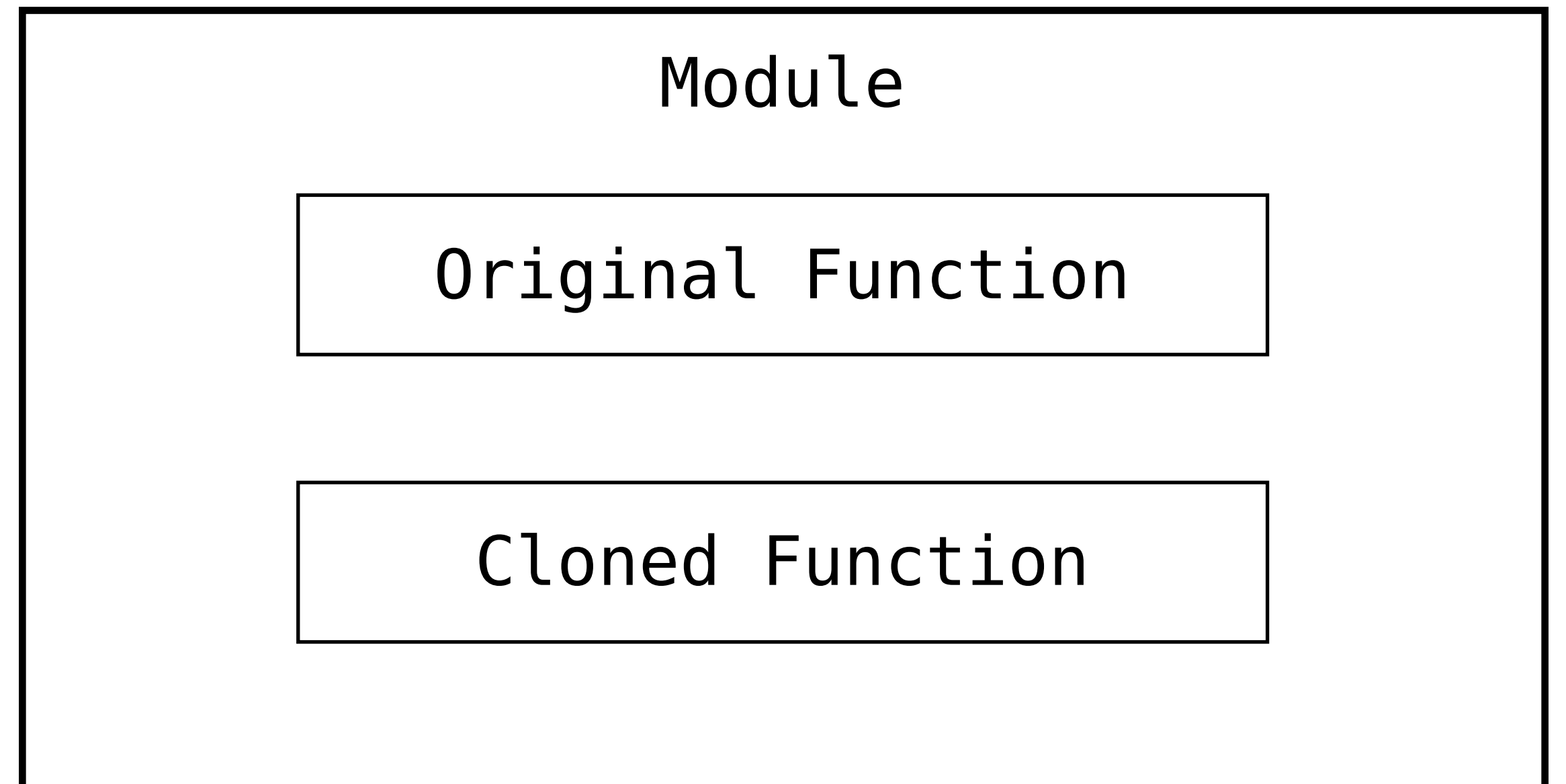
LLVM 3.8

```
Function *  
CloneFunction(const Function *F,  
              ValueToValueMapTy &VMap,  
              bool ModuleLevelChanges,  
              ClonedCodeInfo *CodeInfo);
```



LLVM 3.9+

```
Function *  
CloneFunction(Function *F,  
              ValueToValueMapTy &VMap,  
              ClonedCodeInfo *CodeInfo);
```



Sources vs Binaries

	Precompiled LLVM	LLVM Sources
Fast compile time	✓	✗
Debugging	✗	✓
Asserts	✗	✓

Sources vs Binaries

```
if (EXISTS ${PATH_TO_LLVM}/CMakeLists.txt)
  add_subdirectory(${PATH_TO_LLVM} llvm-build-dir)

  # LLVM_INCLUDE_DIRS ???
  # LLVM_VERSION ???
else()
  ...
endif()
```


Sources vs Binaries

```
if (EXISTS ${PATH_TO_LLVM}/CMakeLists.txt)
  add_subdirectory(${PATH_TO_LLVM} llvm-build-dir)

  get_target_property(LLVM_INCLUDE_DIRS
                      LLVMSupport
                      INCLUDE_DIRECTORIES)

  # LLVM_VERSION ???
else()
  ...
endif()
```

Sources vs Binaries

```
if (EXISTS ${PATH_TO_LLVM}/CMakeLists.txt)
  add_subdirectory(${PATH_TO_LLVM} llvm-build-dir)

  get_target_property(LLVM_INCLUDE_DIRS
                      LLVMSupport
                      INCLUDE_DIRECTORIES)

  string(REGEX MATCH
         "LLVM_VERSION ([0-9]+.[0-9]+.[0-9]+)"
         LLVM_VERSION
         ${PATH_TO_LLVM}/CMakeLists.txt)
else()
  ...
endif()
```

Memory Management

```
std::vector<std::unique_ptr<llvm::Module>> modules;  
LLVMContext context;  
auto module = loadModule("foo.bc", context);  
modules.push_back(std::move(module));
```

Memory Management

```
LLVMContext context;  
std::vector<std::unique_ptr<llvm::Module>> modules;  
auto module = loadModule("foo.bc", context);  
modules.push_back(std::move(module));
```

Memory Management

```
LLVMContext context;  
for (auto x : something) {  
    auto module = loadModule("foo.bc", context);  
    doSomethingWithModule(module);  
    /// the module is destroyed, right?  
}
```

Memory Management

```
LLVMContext context;
for (auto x : something) {
    LLVMContext localContext;
    auto module = loadModule("foo.bc", localContext);
    doSomethingWithModule(module);
    /// the module is destroyed, right? right!
}
```

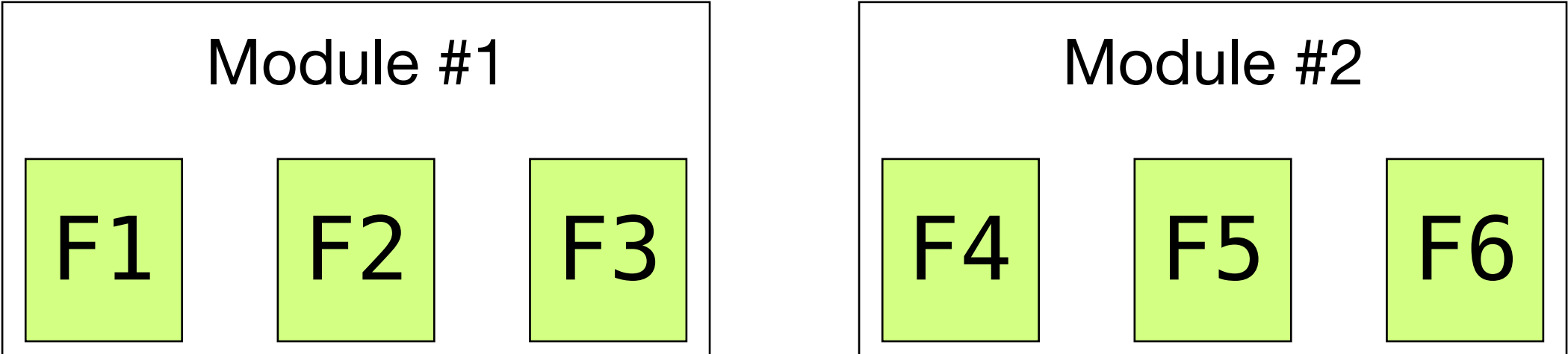
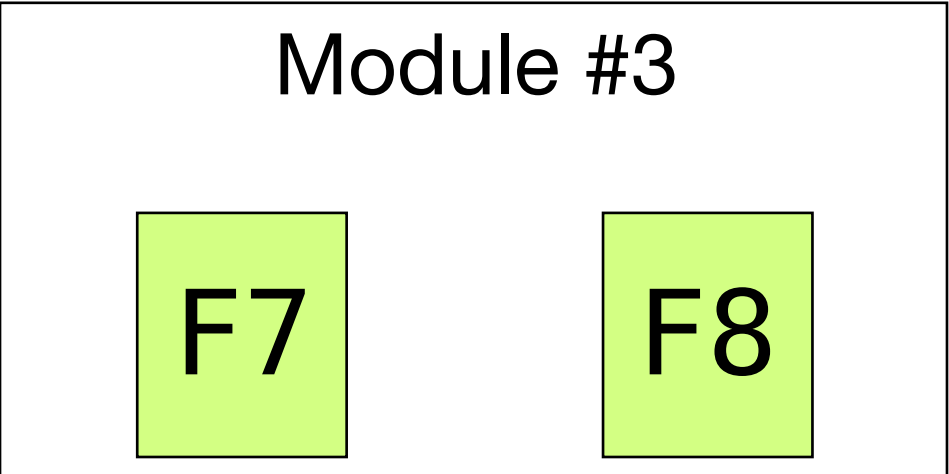
Parallelization



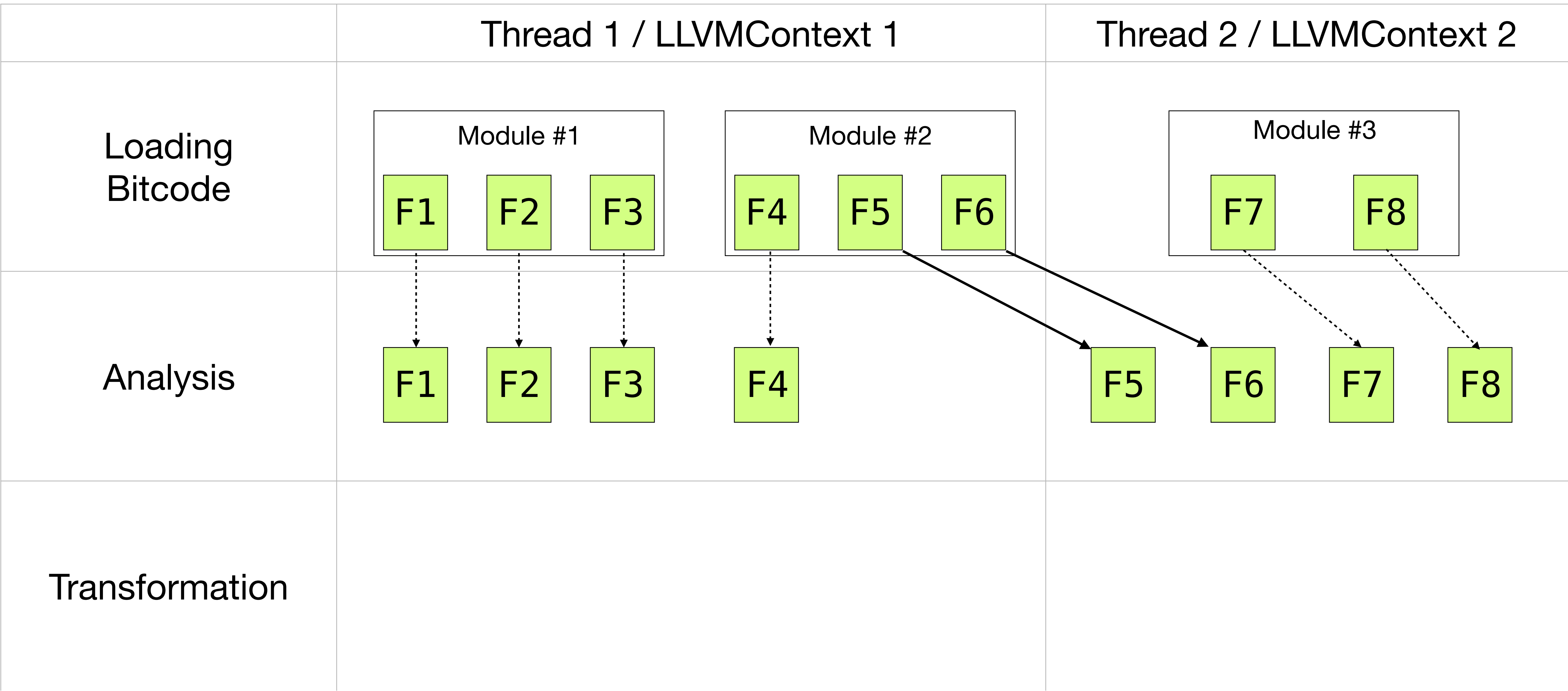
Parallelization

- LLVMContext
 - Module
 - Function
 - Block
 - Instruction
- TargetMachine
 - `orc::SimpleCompiler`
 - CodeGen

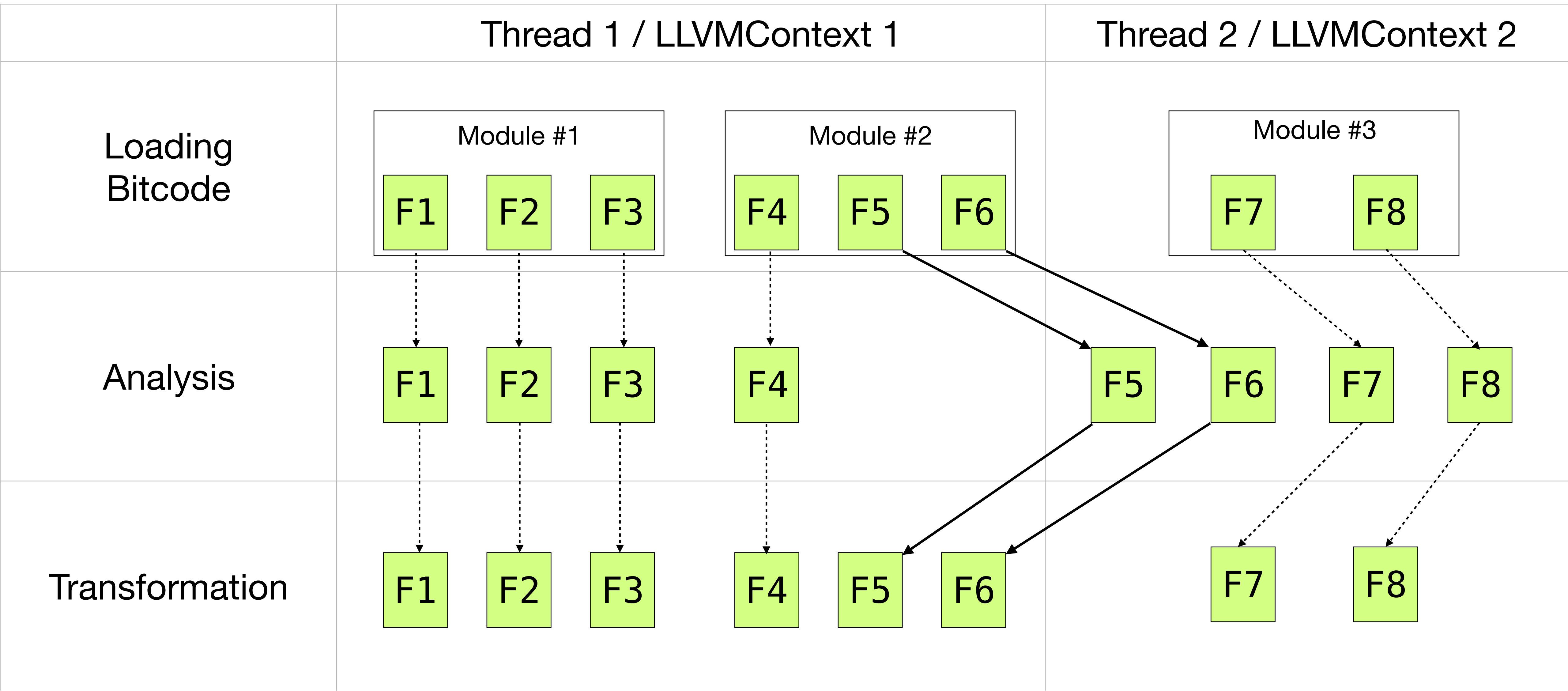
Parallelization

	Thread 1 / LLVMContext 1	Thread 2 / LLVMContext 2
Loading Bitcode	 <p>Module #1</p> <p>F1 F2 F3</p> <p>Module #2</p> <p>F4 F5 F6</p>	 <p>Module #3</p> <p>F7 F8</p>
Analysis		
Transformation		

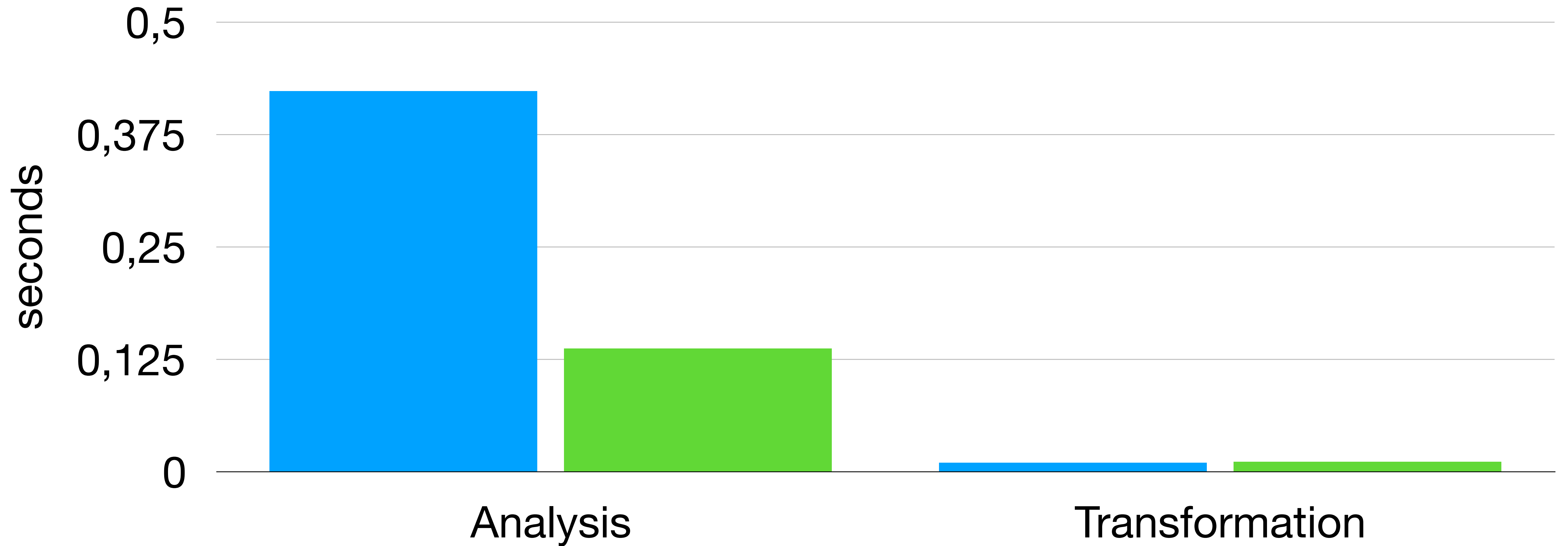
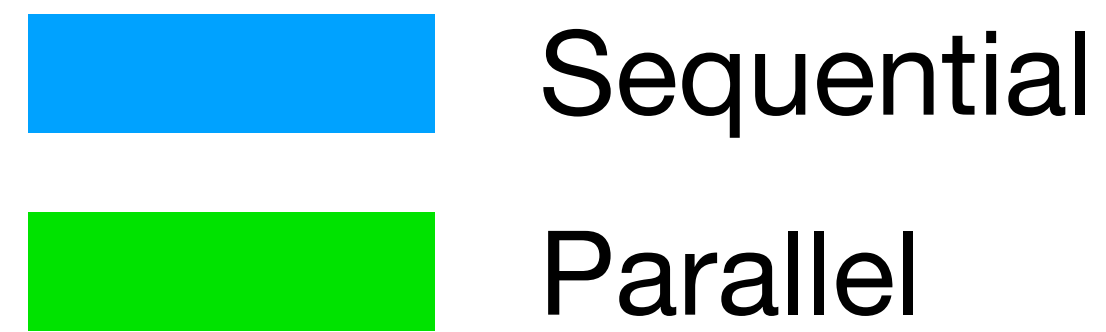
Parallelization



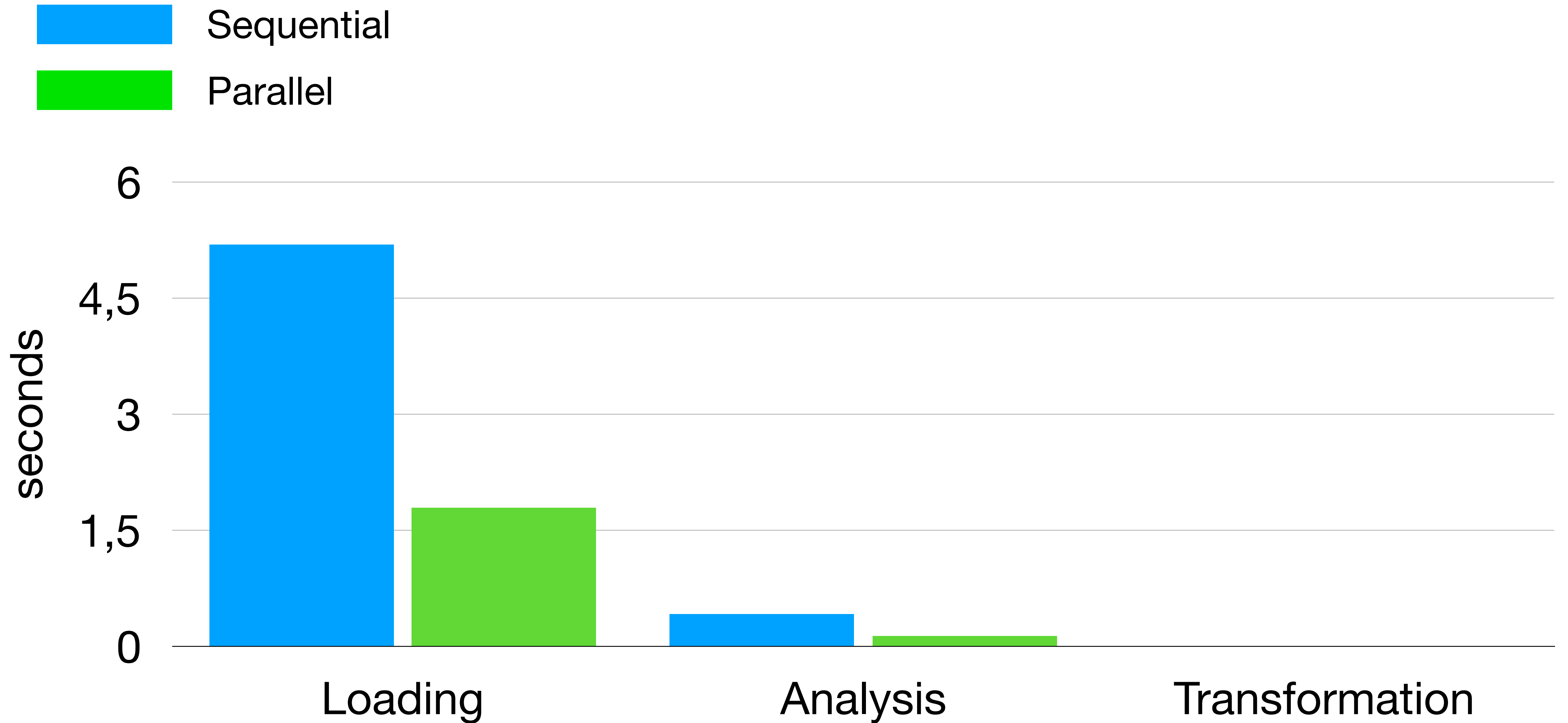
Parallelization



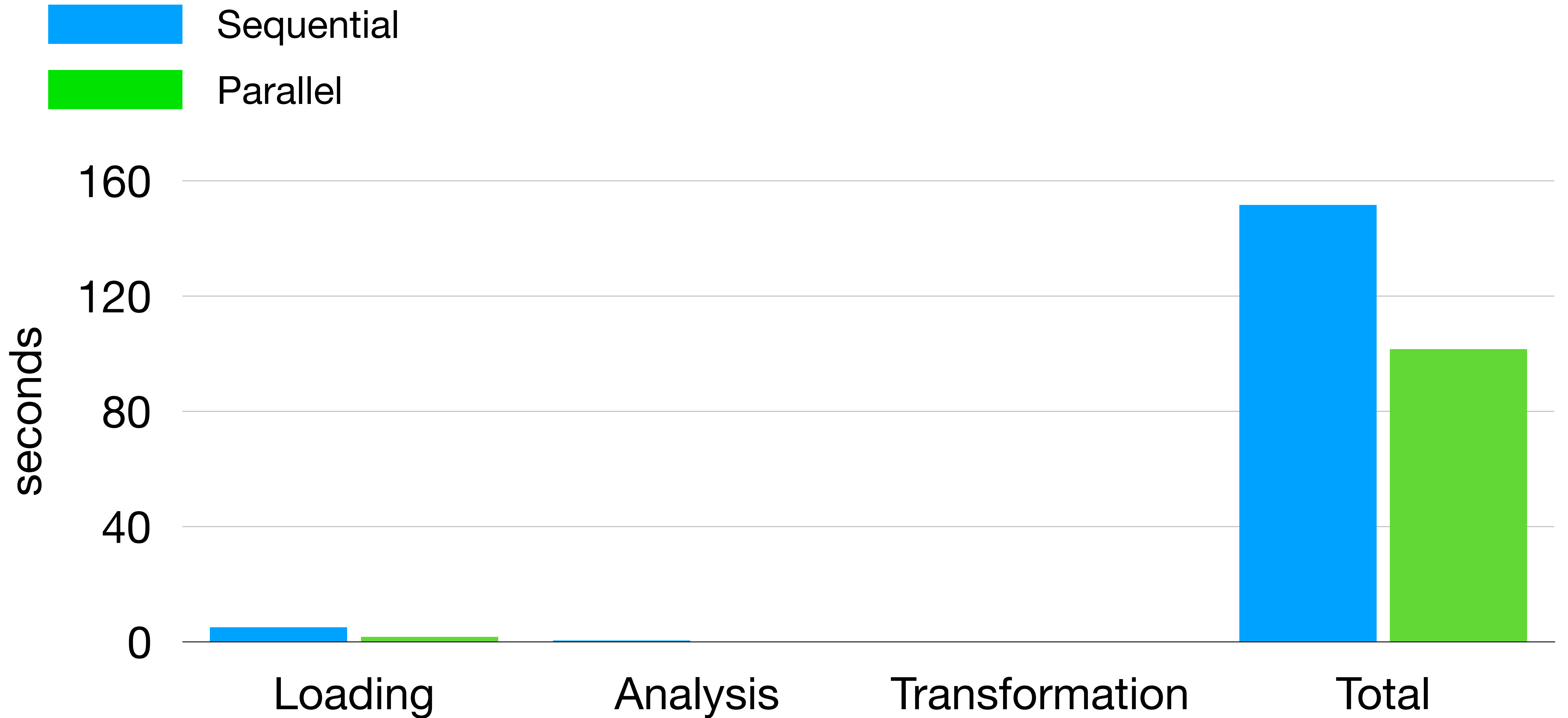
Parallelization



Parallelization



Parallelization



Getting Bitcode

<https://github.com/travitch/whole-program-llvm>

Getting Bitcode

Compiler Flags	Object File	Executable

Getting Bitcode

Compiler Flags	Object File	Executable
<code>-emit-llvm</code>	LLVM Bitcode	N/A

Getting Bitcode

Compiler Flags	Object File	Executable
<code>-emit-llvm</code>	LLVM Bitcode	N/A
<code>-flto</code>	LLVM Bitcode	Machine Code

Getting Bitcode

Compiler Flags	Object File	Executable
<code>-emit-llvm</code>	LLVM Bitcode	N/A
<code>-flto</code>	LLVM Bitcode	Machine Code
<code>-fembed-bitcode</code>	LLVM Bitcode + Machine Code	LLVM Bitcode + Machine Code

-fembed-bitcode

<https://github.com/JDevlieghere/LibEBC>

→ <https://github.com/AlexDenisov/LibEBC>

-fembed-bitcode

```
cmake -G Ninja \  
      -DCMAKE_C_FLAGS=-embed-bitcode \  
      -DCMAKE_CXX_FLAGS=-fembed-bitcode ..
```

```
ninja ADTTests
```

```
ebcutil -e unittests/ADT/ADTTests
```

```
mul-cxx unittests/ADT/ADTTests
```

Multi-OS Support



HashiCorp

Vagrant



ANSIBLE

Multi-OS Support

```
config.vm.define "debian" do |cfg|  
  cfg.vm.box = "debian/stretch64"  
  cfg.vm.provision "ansible" do |ansible|  
    ansible.playbook = "debian-playbook.yaml"  
  end  
end
```

```
config.vm.define "ubuntu" do |cfg|  
  cfg.vm.box = "ubuntu/xenial64"  
  cfg.vm.provision "ansible" do |ansible|  
    ansible.playbook = "ubuntu-playbook.yaml"  
  end  
end
```

Multi-OS Support

tasks:

- name: Prepare Working Directory
- name: Install Required Packages
- name: Install LLVM
- name: Build MuLL
- name: Integration Tests

Multi-OS Support






0.1.0

Edit

 AlexDenisov released this 5 days ago

Initial Release. See README.md for details: <https://github.com/mull-project/mull#usage>

▼ Assets 5

 Mull-0.1.0-LLVM-6.0-debian-9.deb	31.1 MB
 Mull-0.1.0-LLVM-8.0-macOS-10.14.3.dmg	31 MB
 Mull-0.1.0-LLVM-8.0-ubuntu-16.04.deb	33.7 MB
 Source code (zip)	
 Source code (tar.gz)	

Thank you!

Alex Denisov

alex@lowlevelbits.org

https://twitter.com/1101_debian