

# Building LLVM-based tool

Lessons learned

# whoami

- Alex Denisov
- Software Engineer at PTScientists GmbH
- LLVM Hacker
- <https://lowlevelbits.org>
- [https://twitter.com/1101\\_debian](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=YEgiyilCkpQ>
- Mull it over: mutation testing based on LLVM  
<https://ieeexplore.ieee.org/document/8411727>
- Works on Linux, macOS, FreeBSD
- Works with LLVM 3.9 - 7.0

# 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 --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.
```

# CMake

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

# CMake

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

# 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

```
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

```
▼ 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/↵
```

# Multiple LLVM Versions

LLVM 7.0

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

namespace llvm_compat {
    using namespace llvm;

    typedef JITSymbolResolver SymbolResolver;
    typedef JITSymbol JITSymbolInfo;
    typedef JITSymbol JITSymbol;

    uint64_t JITSymbolAddress(JITSymbol &symbol);

    JITSymbolFlags JITSymbolFlagsFromObjectSymbol(
        const object::BasicSymbolRef &symbol);

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

LLVM 3.9

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

namespace llvm_compat {
    using namespace llvm;

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

    uint64_t JITSymbolAddress(JITSymbol &symbol);







    JITSymbolFlags JITSymbolFlagsFromObjectSymbol(
        const object::BasicSymbolRef &symbol);

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

# Multiple LLVM Versions

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

# 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-dir)

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



# Sources vs Binaries

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

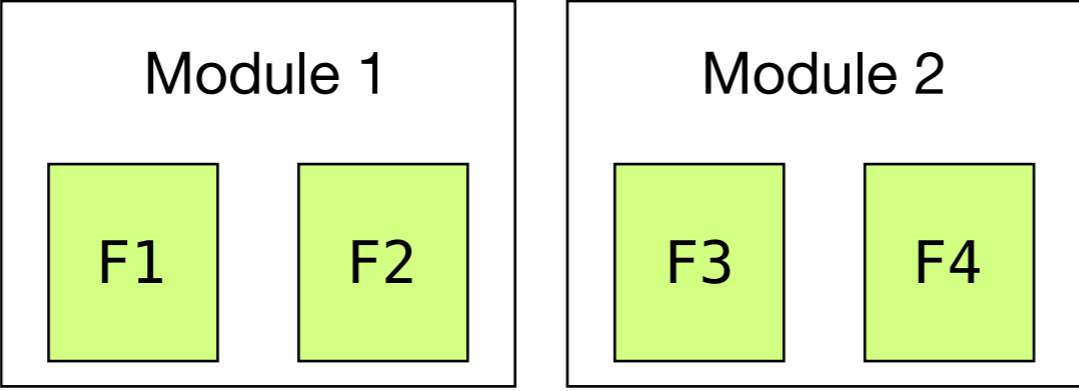
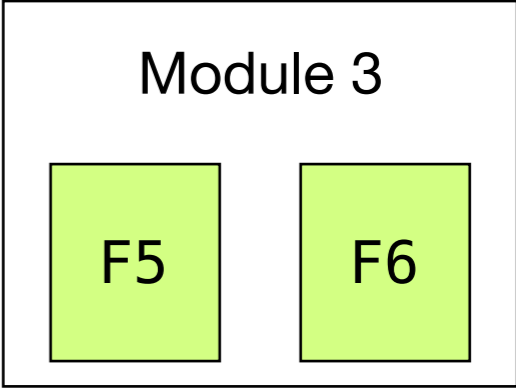
# Parallelization

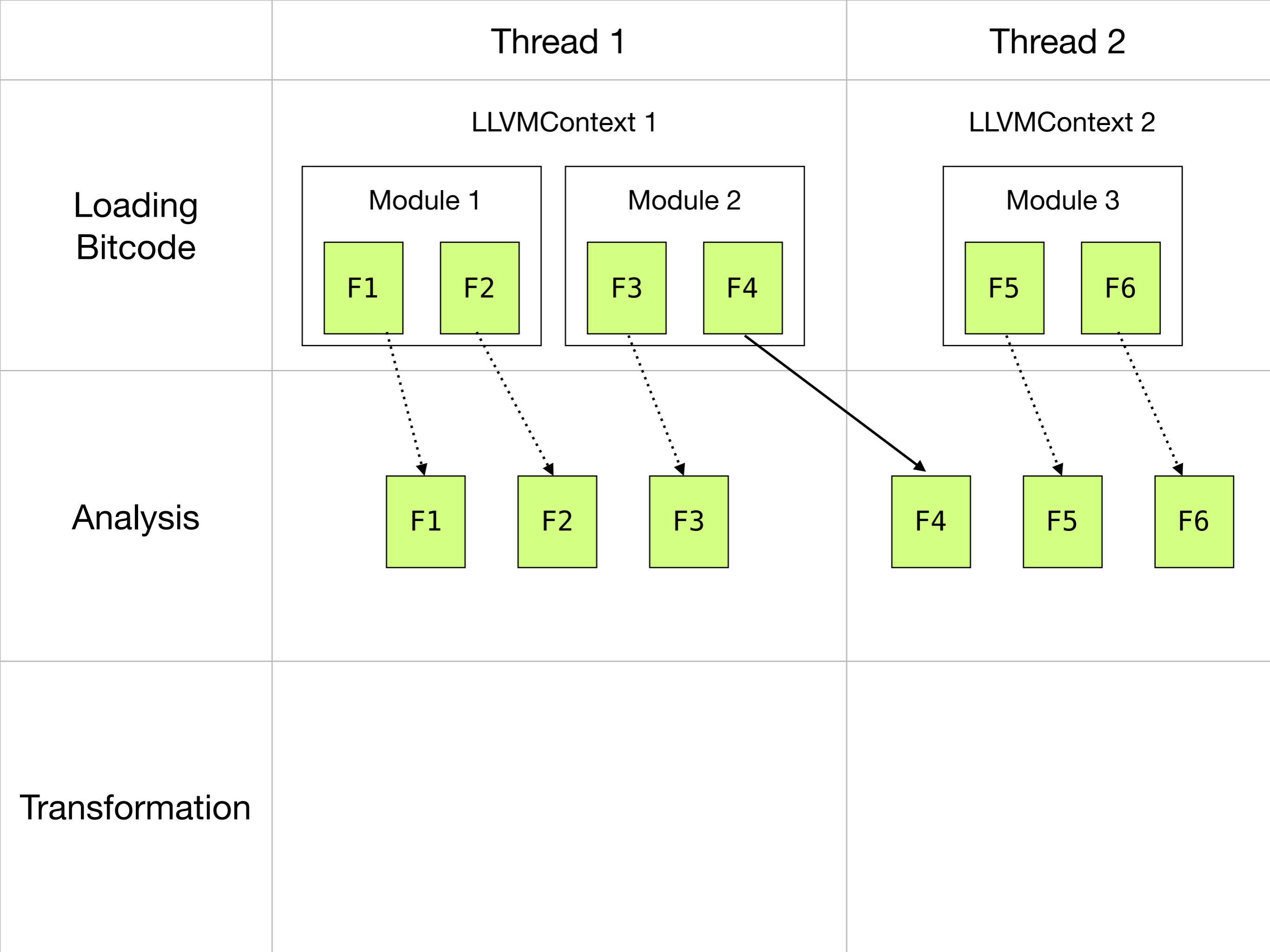


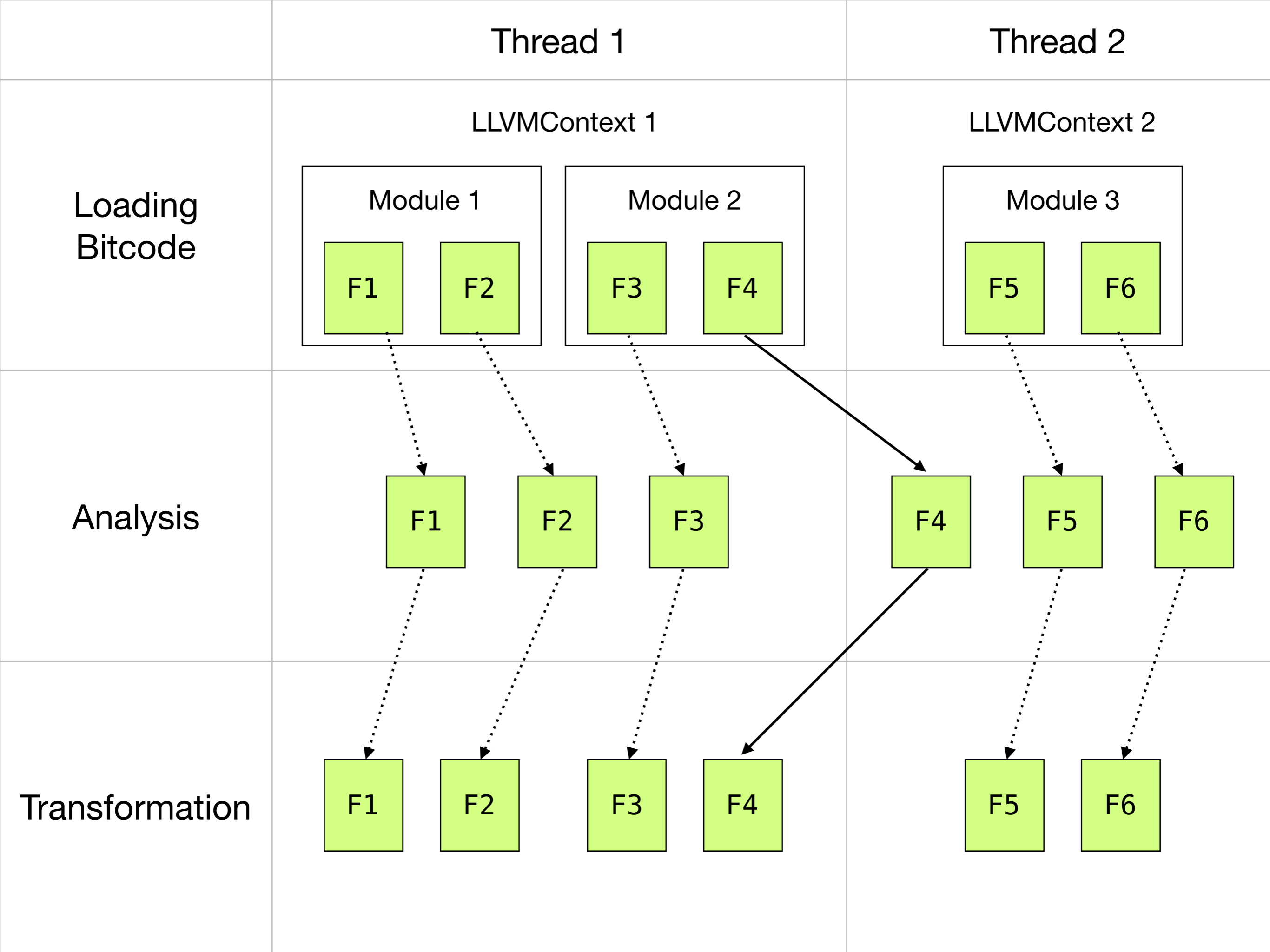
# Parallelization

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

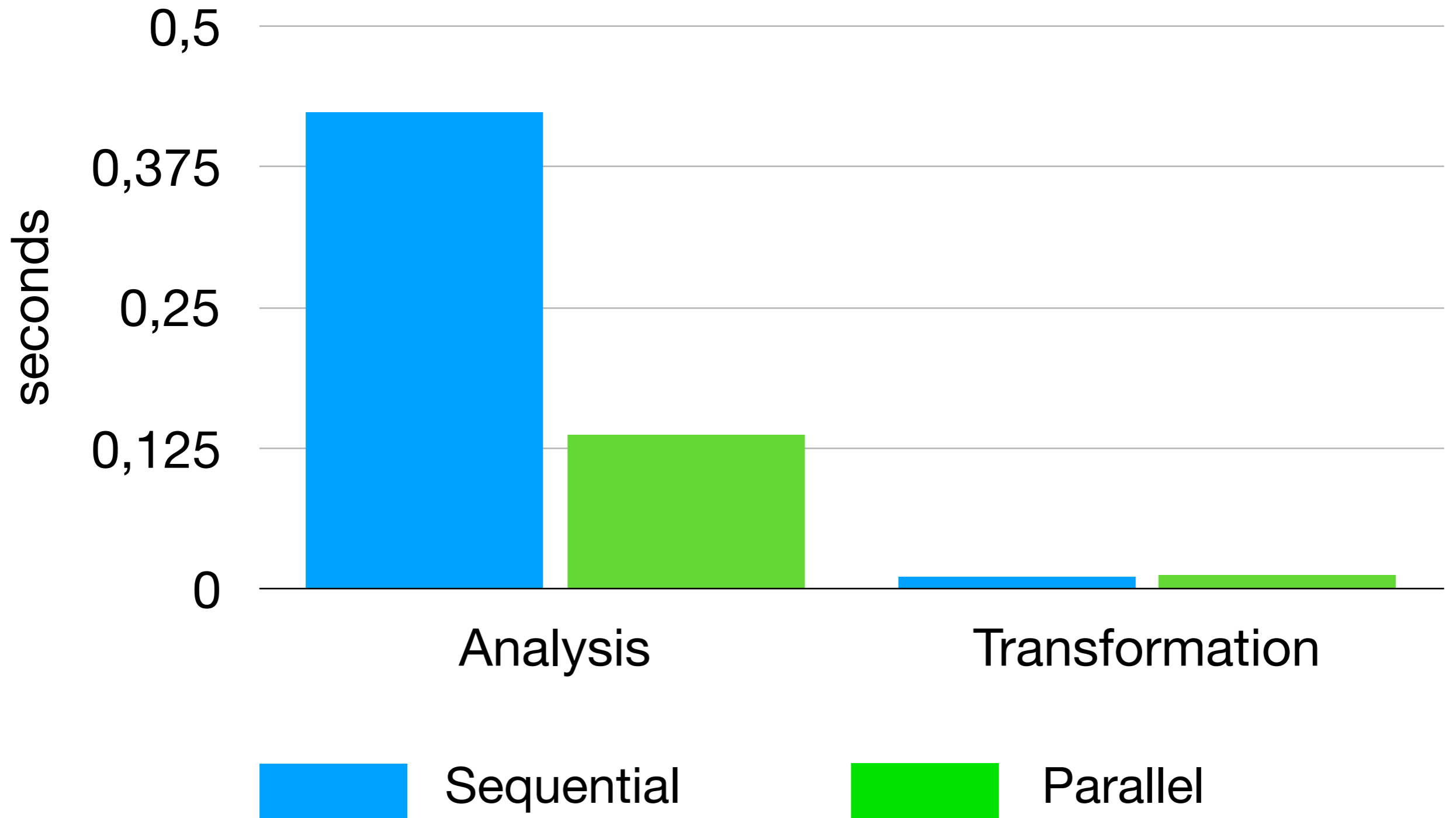


	Thread 1	Thread 2
Loading Bitcode	<p>LLVMContext 1</p>  <p>Module 1</p> <p>Module 2</p> <p>F1 F2 F3 F4</p>	<p>LLVMContext 2</p>  <p>Module 3</p> <p>F5 F6</p>
Analysis		
Transformation		

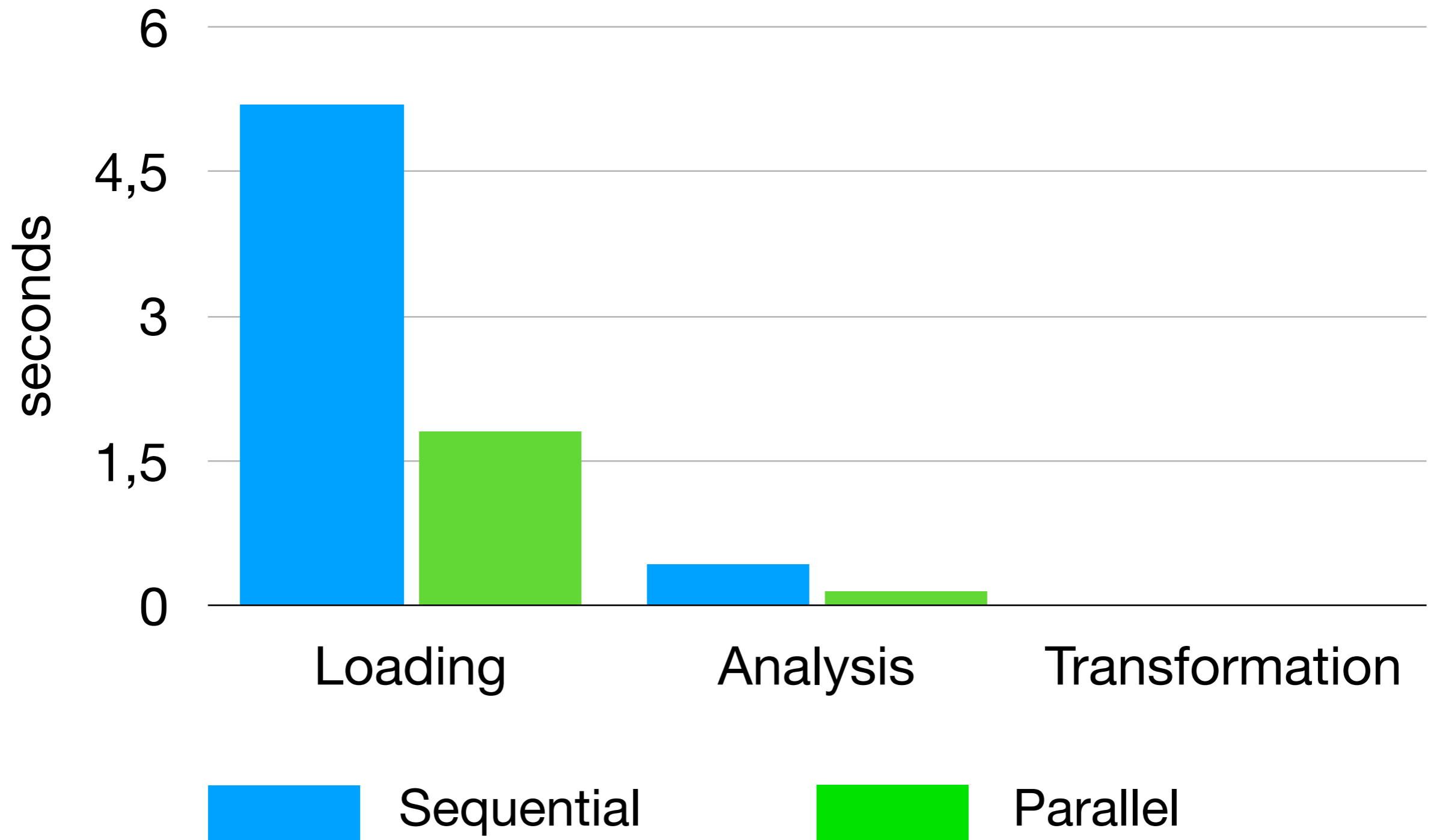




# Parallelization



# Parallelization



# Getting Bitcode

Compiler Flags	Object File	Executable

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



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

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
```

# -fembed-bitcode

```
> cmake -G Ninja \  
        -DCMAKE_C_FLAGS=-embed-bitcode \  
        -DCMAKE_CXX_FLAGS=-fembed-bitcode ..  
> ninja ADTTests  
> mull-cxx unittests/ADT/ADTTests
```

# Multi OS Support



HashiCorp

**Vagrant**



ANSIBLE

# Vagrant

```
config.vm.define "debian" do |m|  
  m.vm.box = "debian/stretch64"  
  
  m.vm.provision "ansible" do |a|  
    a.playbook = "debian.yaml"  
  end  
end  
  
config.vm.define "ubuntu" do |m|  
  m.vm.box = "ubuntu/xenial64"  
  
  m.vm.provision "ansible" do |a|  
    a.playbook = "ubuntu.yaml"  
  end  
end
```

# Vagrant

- > `vagrant up debian`
- > `vagrant ssh debian`
- > `vagrant destroy debian`



# Ansible

```
tasks:-  
  - name: Install Required Packages  
    apt:  
      name: "{{ packages }}"  
      state: present  
    become: true  
  
  - name: Install LLVM  
    include: helpers/download-llvm.yaml  
  
  - name: Build MULL  
    include: helpers/build-mull.yaml  
  
  - name: Integration Tests  
    include: helpers/integration-tests.yaml
```

# Ansible

- > `ansible-playbook localhost debian.yaml`
- > `ansible-playbook localhost freebsd.yaml`
- > `ansible-playbook localhost macos.yaml`

# Thank you

Alex Denisov

[alex@lowlevelbits.org](mailto:alex@lowlevelbits.org)

[https://twitter.com/1101\\_debian](https://twitter.com/1101_debian)