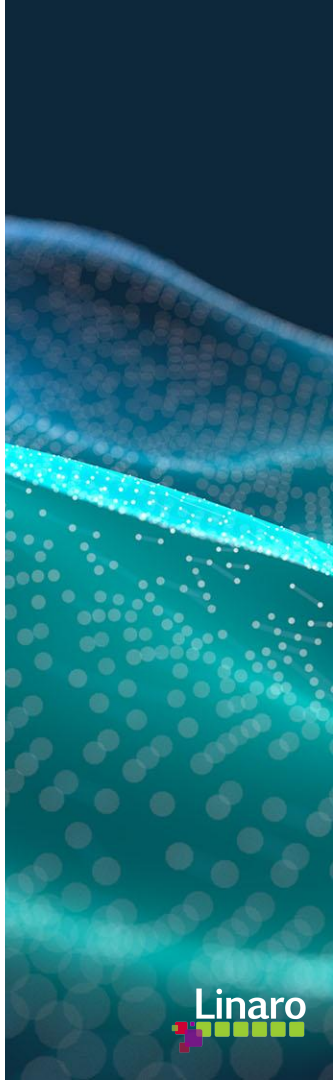
LLVM Testsuite Under The Hood

Muhammad Omair Javaid
Senior Engineer, Linaro Ltd

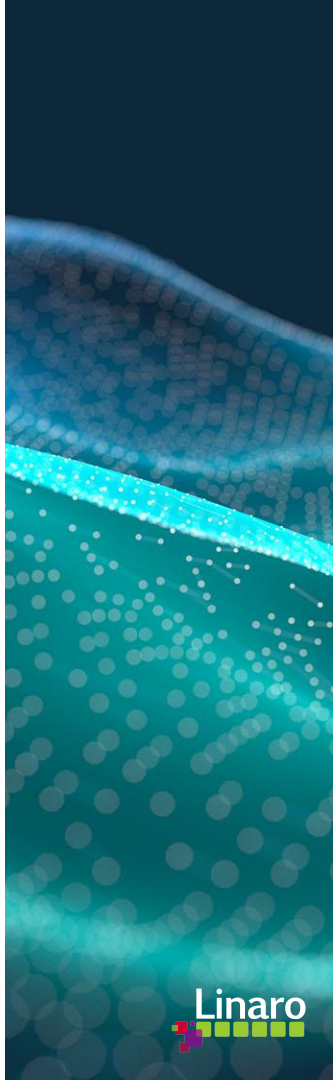
Who Am I and Who is Linaro?

- Who Am I?
  - Toolchain Engineer with core expertise in debug technologies
  - Contribution in GDB, LLDB and OpenOCD
  - Linaro's project lead for LLDB, Flang and Windows on Arm toolchain
  - LLVM release manager for Arm/AArch64 Linux/Windows
- Who is Linaro?
  - Linaro is a collaborative engineering organization
  - Leads collaboration on open-source software development for Arm
  - Contributes to Linux kernel, GNU/LLVM Toolchains and many more...
  - www.linaro.org



Agenda for the Next 20 Minutes

- What and Why of LLVM Testsuite?
- LLVM Testsuite vs LLVM Project Tests: What is Different?
- Tools and Technology used by LLVM Testsuite
- LIT and CMake: How LLVM Testsuite Works?
- LLVM Test Suite Repository: A Peek Inside..
- Our Mission: LLVM Testsuite enablement for Windows



What is LLVM Testsuite?

- The llvm-test-suite is hosted at https://github.com/llvm/llvm-test-suite.git
- Houses a variety of tests
  - Singe/Multi source whole programs, real-world applications
  - Written in high-level languages like C, C++, Fortran or even in LLVM Bitcode
  - Benchmarks, Regression and Unit Tests
  - Supports importing external source code repositories
- These tests serve multiple purposes
  - Validate LLVM's correctness
  - Benchmark LLVM's performance
  - Exports performance results to visualize in LNT
  - Support various build configurations with different optimization levels

Why segregate llvm-test-suite from llvm-project?

- Nature of Tests
  - llvm-project
    - Tests tightly coupled with LLVM source code
    - Conforming to LLVM license and coding standards
    - Primarily focused on correctness validation
      - Unit tests that test LLVM code
      - Regression tests that validates output
  - llvm-test-suite
    - Real-world programs
    - Generates a variety of performance metrics
    - Variety of licenses coding standards

Why segregate llvm-test-suite from llvm-project?

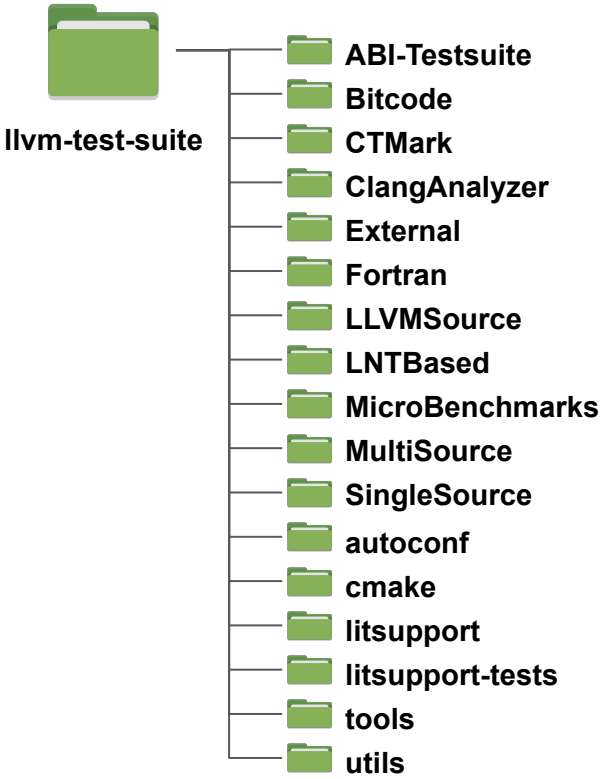
- Clutter Reduction in *llvm-project*
  - Reduced Repository Size
  - Optimized Build and Test Times
  - Reduce maintenance burden
- Collaboration and Scalability in llvm-test-suite
  - Room for Expansion
    - Import of external code-bases
    - Easier integration of tools like perf
  - Licensing Flexibility
    - Can host GPL, MIT, Apache etc under one roof

Performance Metrics emitted by LLVM Testsuite

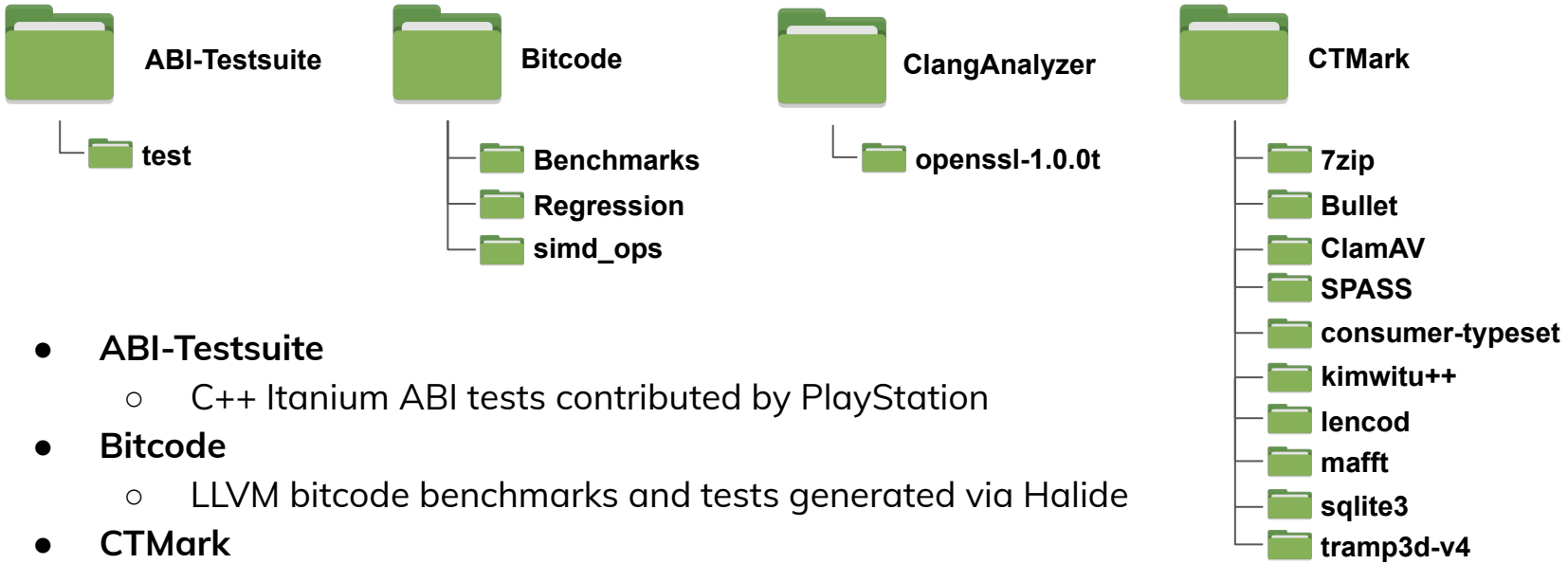
- Compilation correctness
- Code Size
- Compilation Speed
- Linking Speed
- Output Correctness
- Running Time
- Benchmark Score
- Performance Data for LNT server
- Generates Perf reports

LVM Test Suite Repository: A Peek Inside

- llvm-test-suite source tree



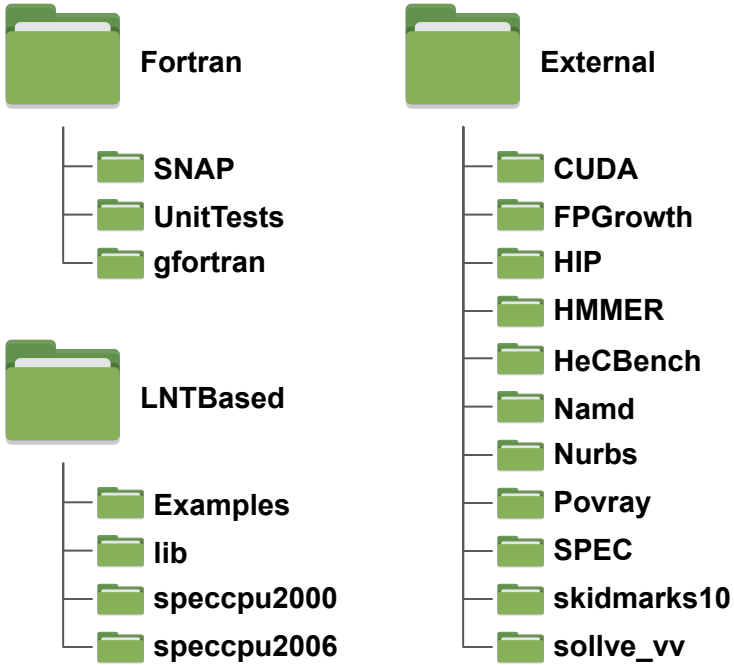
LLVM Test Suite Repository: A Peek Inside



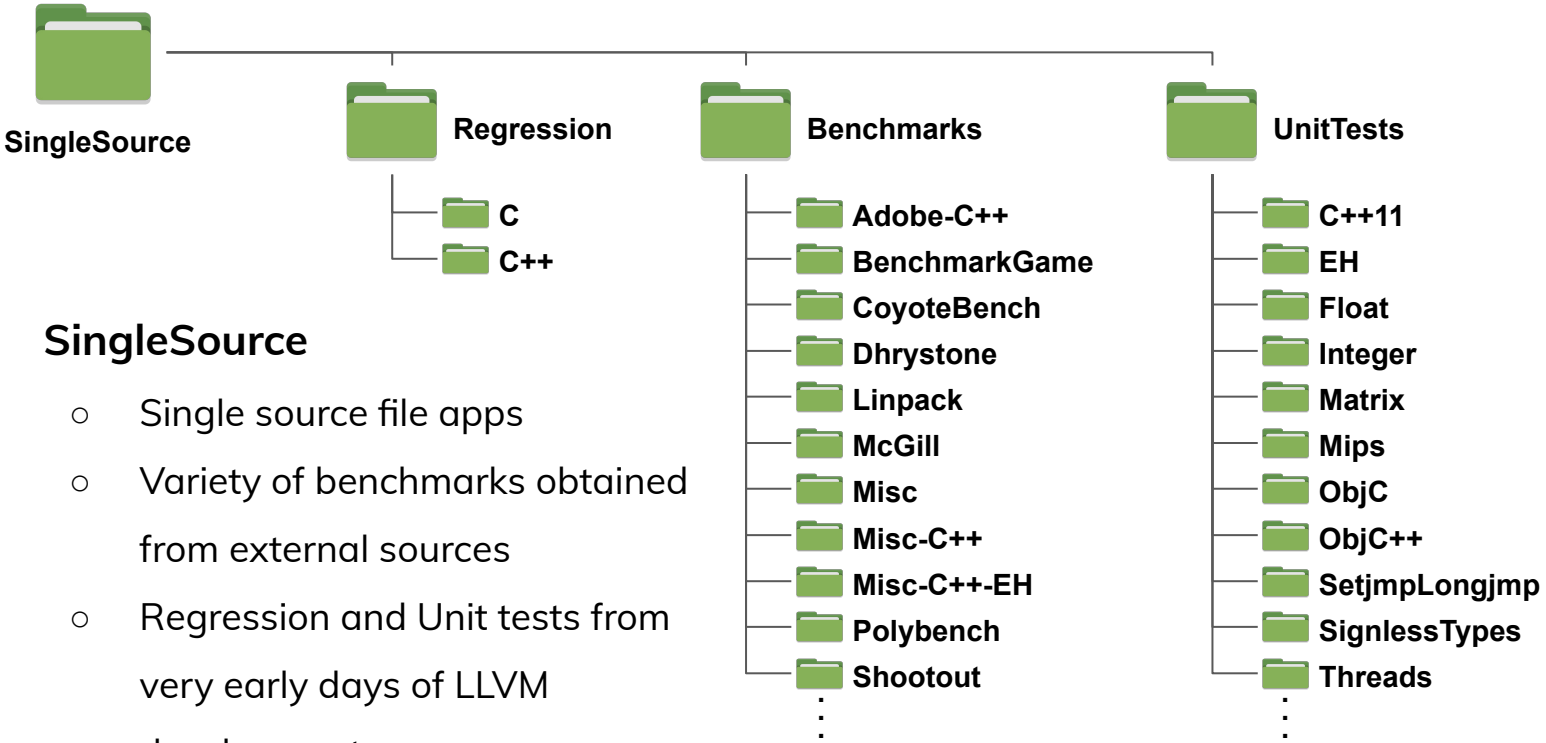
- **ABI-Testsuite**
  - C++ Itanium ABI tests contributed by PlayStation
- **Bitcode**
  - LLVM bitcode benchmarks and tests generated via Halide
- **CTMark**
  - Compile time tracking using long compile time applications
- **ClangAnalyzer**
  - Benchmark and test the clang static analyzer

LLVM Test Suite Repository: A Peek Inside

- **Fortran**
  - Includes SNAP application gfortran, and NIST test suites.
- **LNTBased**
  - Tests written as LNT NT(nightly test) Python modules.
  - Return an LNT testing report based on user parameters.
- **External**
  - Provides configuration data to imports Benchmarks and tests from external source into LLVM Testsuite

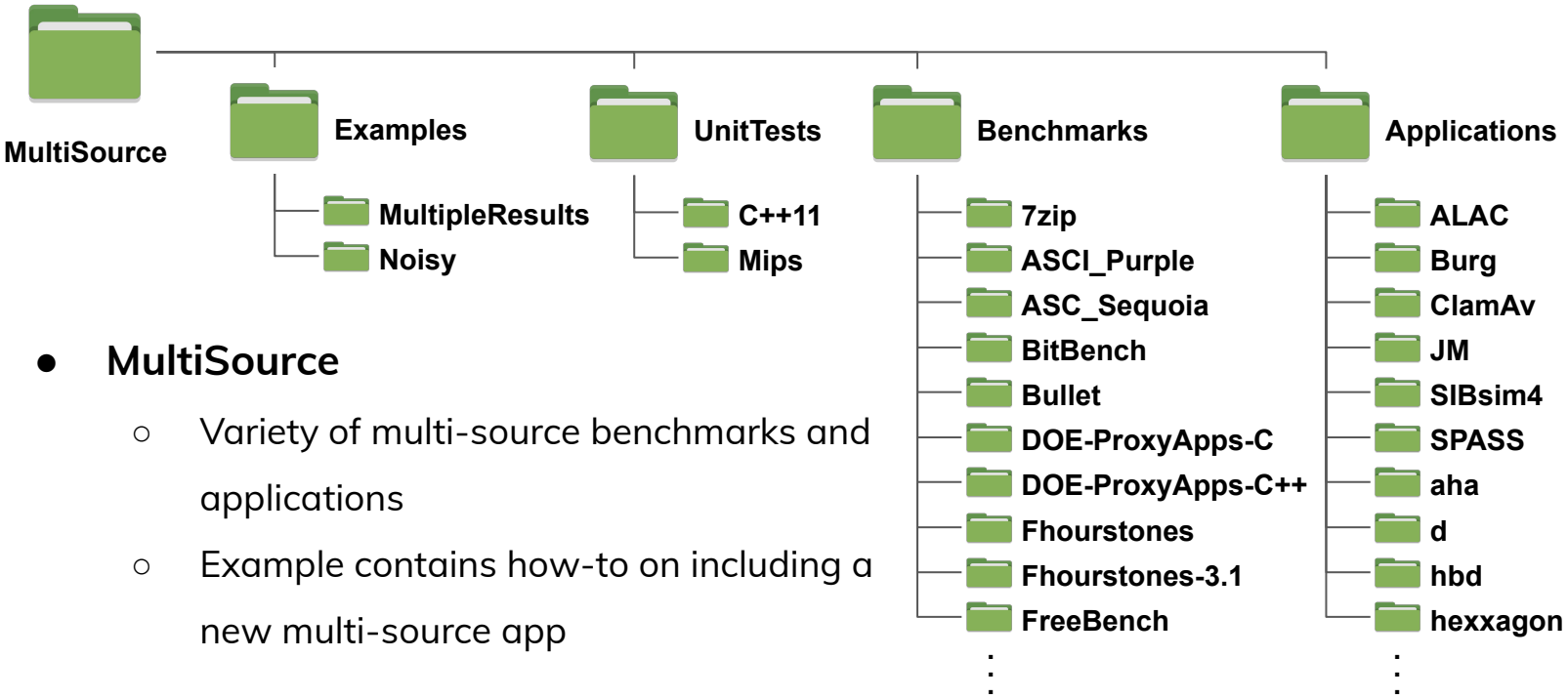


LLVM Test Suite Repository: A Peek Inside



- **SingleSource**
  - Single source file apps
  - Variety of benchmarks obtained from external sources
  - Regression and Unit tests from very early days of LLVM development.

LLVM Test Suite Repository: A Peek Inside



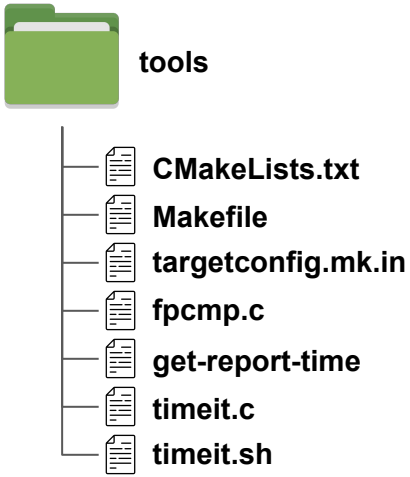
- **MultiSource**
  - Variety of multi-source benchmarks and applications
  - Example contains how-to on including a new multi-source app

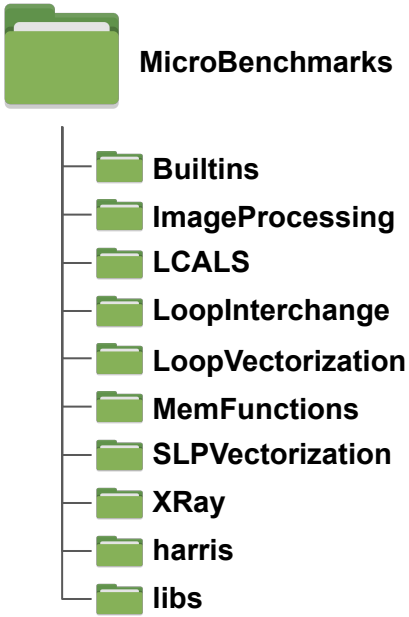
Linaro

LLVM Test Suite Repository: A Peek Inside

- **Tools**
  - **timeit** utility which forks compilation, linking and execution and calculates time
  - **fpcmp** compares execution output against reference output for correctness validation.

- **MicroBenchmarks**
  - Benchmarks based on google benchmark library

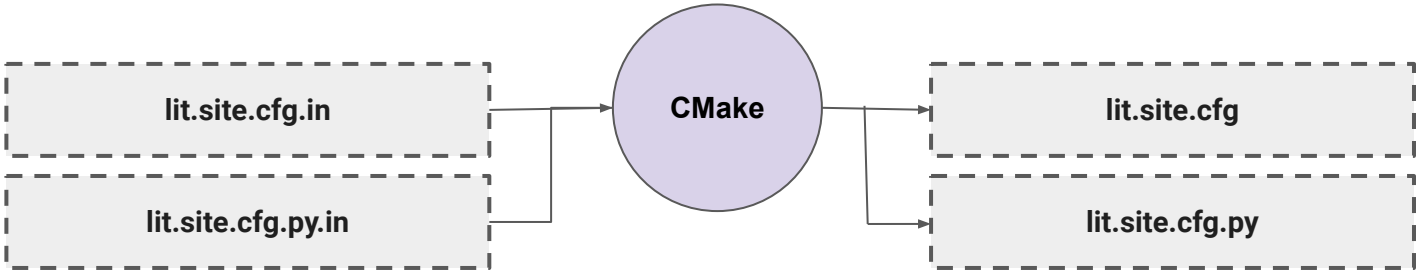




Anatomy of basic CMake + LIT Testsuite

- Processing by CMake **lit.site.cfg.in** or **lit.site.cfg.py.in**
  - CMake fills placeholders based on the build environment
  - Helps out-of-tree test execution
- Invoke llvm-lit with a path to test folder or build directory
  - bin/llvm-lit [test or test suite path]
- LIT locates a valid **lit.site.cfg** or **lit.site.cfg.py**
  - Apply test suite configuration from lit.cfg or lit.cfg.py file.
  - Recursively searches for tests files and prepares a list of tests
- Runs tests based on **test_format** found in configuration file
- Report tests results after running tests in the test directory

Anatomy of basic CMake + LIT Testsuite

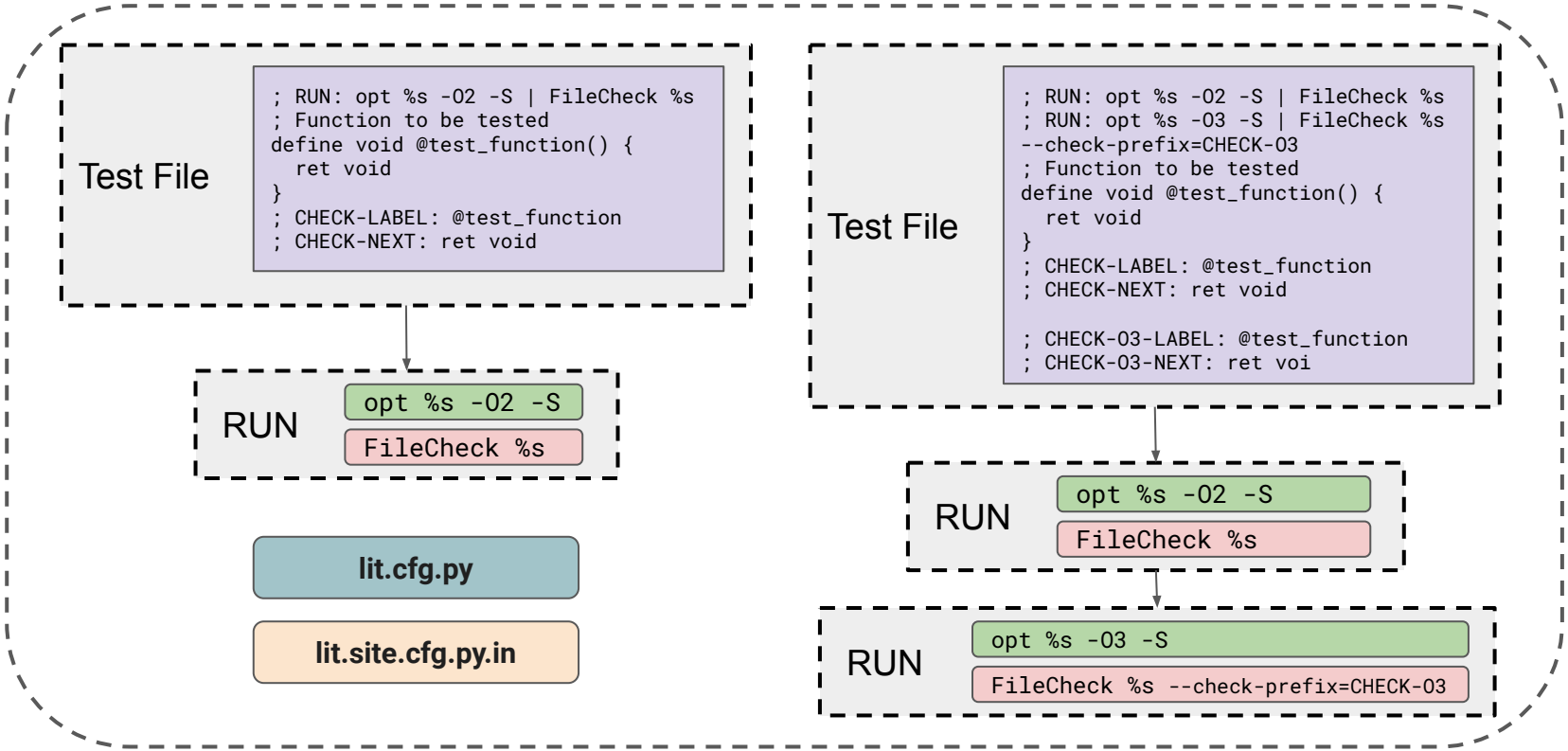



```
config.test_source_root = _directory
config.test_exec_root = _directory
config.remote_client = "@TEST_SUITE_REMOTE_CLIENT@"
config.remote_host = "@TEST_SUITE_REMOTE_HOST@"
config.run_under = "@TEST_SUITE_RUN_UNDER@"
config.user_mode_emulation =
@TEST_SUITE_USER_MODE_EMULATION@
config.strip_tool = "@CMAKE_STRIP@"
config.profile_generate = @TEST_SUITE_PROFILE_GENERATE@
config.llvm_profdata = "@TEST_SUITE_LLVM_PROFDATA@"
config.test_modules = "@LIT_MODULES@".split(";")
```

```
config.test_source_root = _directory
config.test_exec_root = _directory
config.remote_client = "ssh"
config.remote_host = ""
config.run_under = ""
config.user_mode_emulation = False
config.strip_tool = "/usr/bin/strip"
config.profile_generate = False
config.llvm_profdata = ""
config.test_modules =
"run;hash;compiletime;timeit".split(";")
```

Linaro

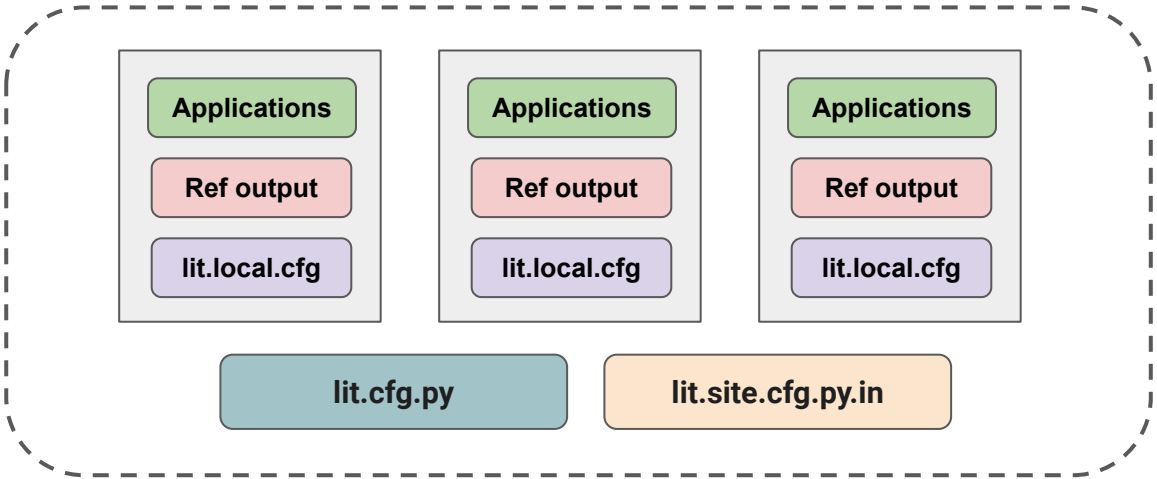
Anatomy of basic CMake + LIT Testsuite



Linaro

LIT and CMake: How LLVM Testsuite Works?

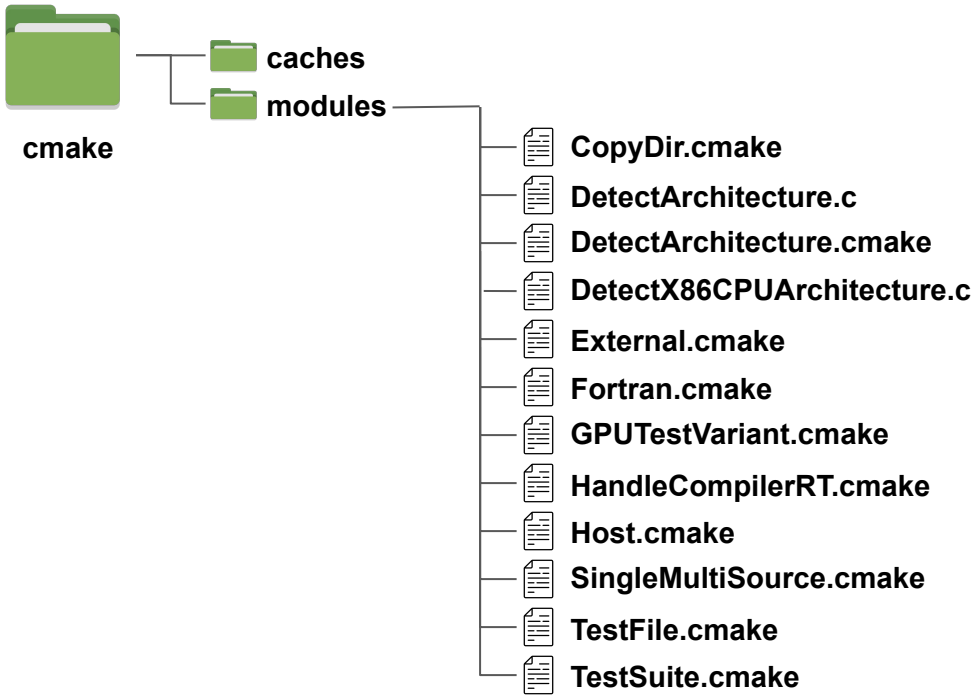
LLVM Testsuite source code before CMake invocation



LIT and CMake: How LLVM Testsuite Works?

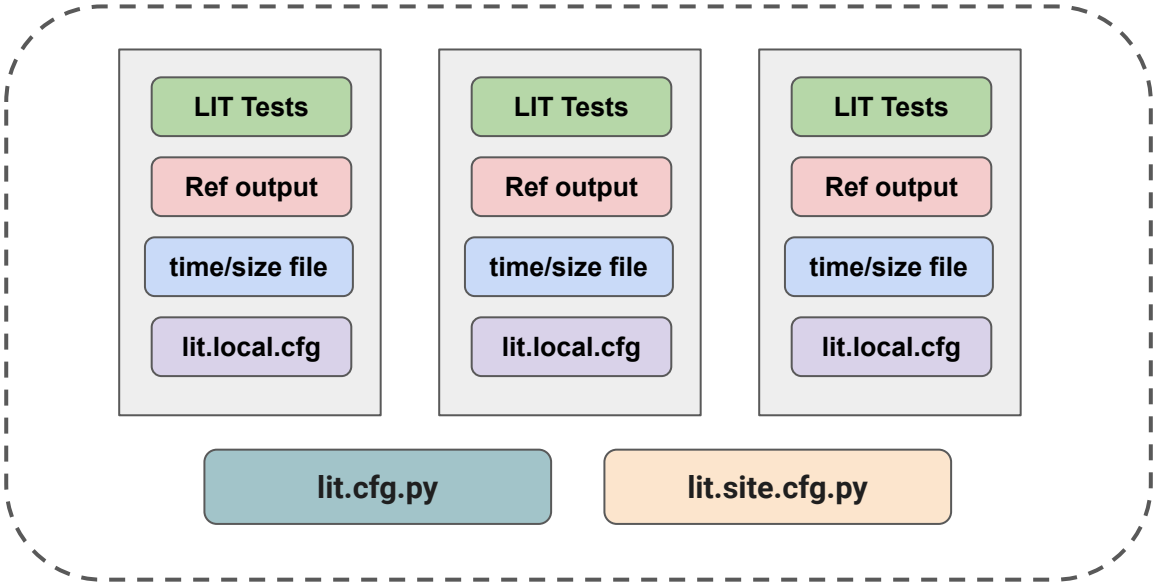
cmake → modules

- Generates LIT test files for each application included in current build.
- Test files are created in build folder
- Symlinks are created in builds folder for reference output
- Wraps compilation and linking around **timeit** for time file generation
- Add build step for size file generation using llvm-size



LIT and CMake: How LLVM Testsuite Works?

CMake generates LIT tests in build directory



LIT and CMake: How LLVM Testsuite Works?

LIT **config.test_format**

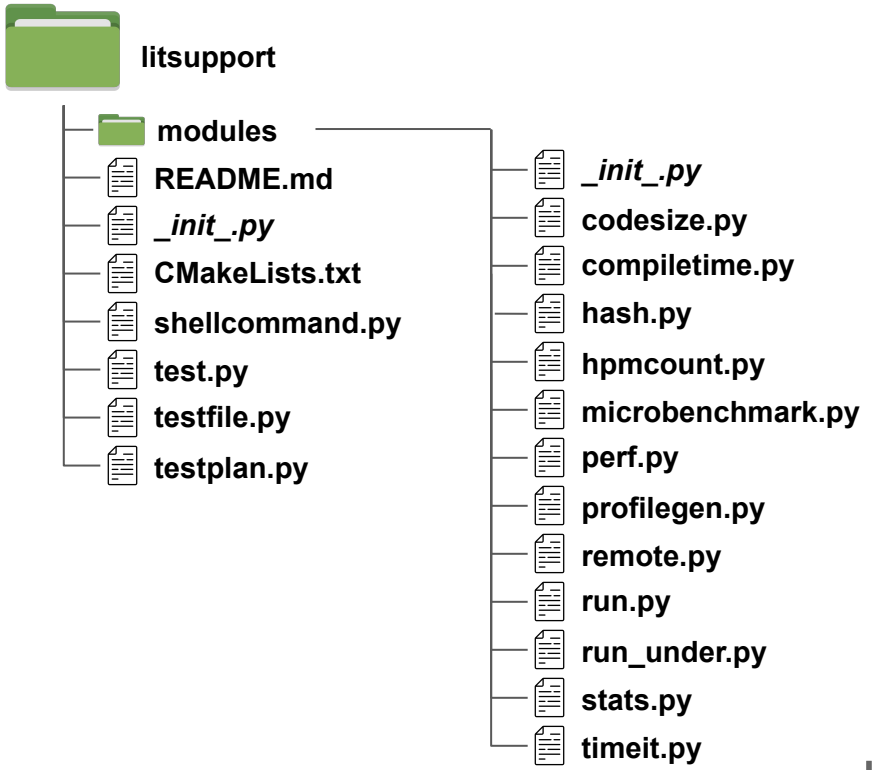
- Extend LIT functionality by introducing custom test_format

litsupport.test.TestSuiteTest()

- Inherits from LIT builtin shell format

litsupport and **litsupport → modules**

- Extend LIT functionality
- Executes tests under **timeit** or **perf**
- Generates time files



Linaro

LLVM Testsuite enablement for Windows

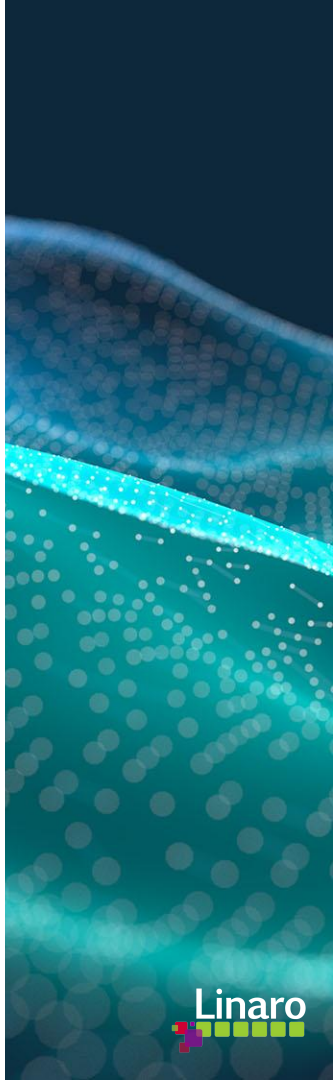
- Mission → Benchmark clang-cl and flang-new on windows
  - All tests should be able to compile
  - Generate performance metrics similar to Linux
  - LNT should be able to execute LLVM Testsuite on Windows
  - Integration with Windows Perf tools
- Steps in progress
  - Porting of timeit and fpcmp
  - Tweaking of CMake module to enable on Windows
  - Porting to TestSuite LIT test_format to run on Windows
  - Porting of test various applications to support clang-cl and cl
- Future Goals
  - Expand test coverage and port more test applications
  - Integration with LNT
  - Integration with Windows Perf

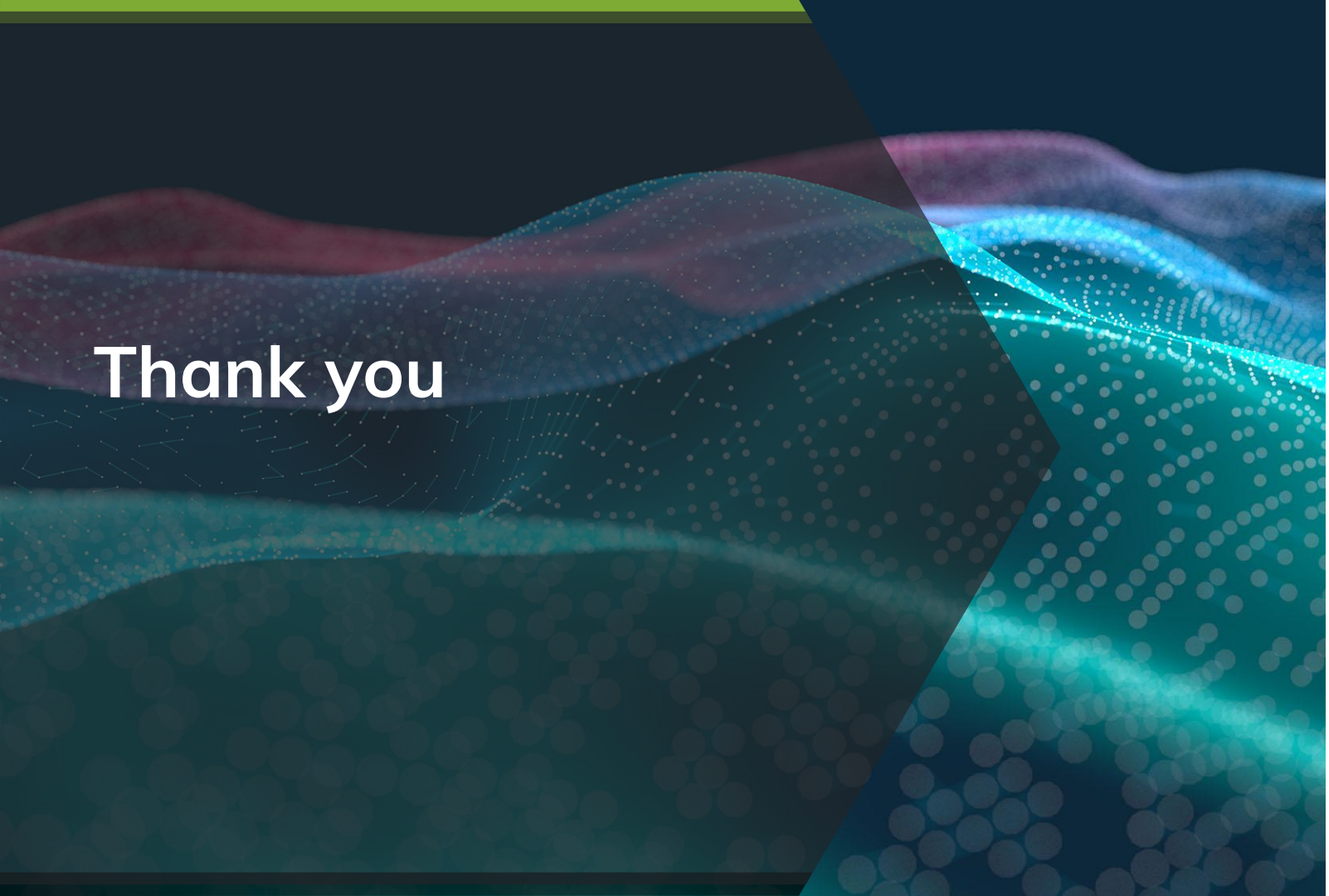
Linaro

LLVM Testsuite enablement for Windows

- **Questions?**

- **Contact:** omair.javaid@linaro.org





Linaro