To be able to decrease significantly the build time, you need to use ccache with your Axmol project. The tool ccache is a compiler cache that can drastically improve the build for C and C++ projects and it is hosted here (https://ccache.dev)

Before implementing this solution, it was taking us about 14 to 15 minutes to build each time that we changed a single line of code. Everything was rebuild at each time. Obviously is not acceptable in a professional/production environment.

After implementing this solution, it takes us now about 1min30s to 2min to build (depending on the change). That's about 8 times faster than before. Now our development time with Axmol is much faster and acceptable. We are back into business.

These are the steps to configure your system with ccache.

Step 1) Install ccahe:

a) On Mac: use brew to install ccache. It's very simple. From your terminal just enter the command:

```
brew install ccache
```

b) On Linux: you just install the ccache package. Any distribution out there should have the package available.

b) On Windows: I don’t have a Windows system so I’m not sure. But I do believe that you can find the instructions on ccache website (https://ccache.dev).

Step 2) create a test file ccache-clang script file:
Create a small script which will redirect the compile command to `ccache` and use the `CC` and `CXX` attributes to tell Xcode about the script. Add these lines in your “ccache-clang” file.

```
#!/bin/sh
export PATH=$PATH:/usr/local/bin
if type -p ccache >/dev/null 2>&1; then
  export CCACHE_MAXSIZE=20G
  export CCACHE_CPP2=true
  export CCACHE_HARDLINK=true
  export CCACHE_SLOPPINESS=file_macro,time_macros,include_file_mtime,include_file_ctime,file_stat_matches

  exec ccache /usr/bin/clang "$@"
else
  exec clang "$@"
fi
```

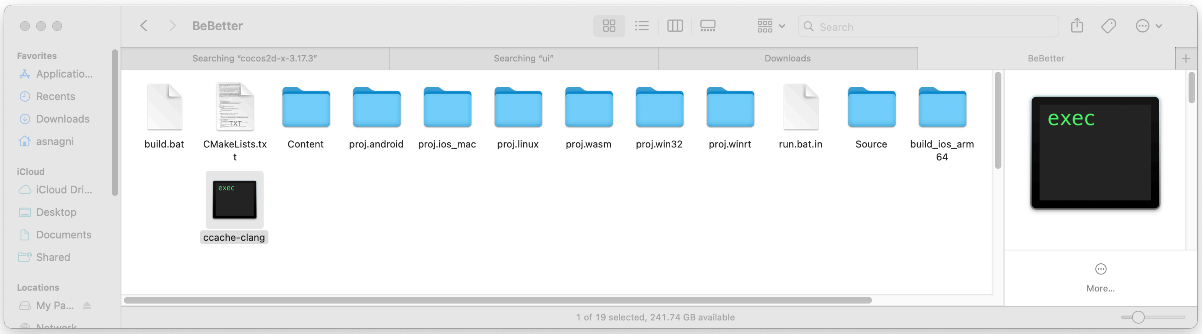
You can find the signification of these parameters here and adjust them for your case if needed (https://ccache.dev/manual/3.2.1.html)

* CCACHE_MAXSIZE: sets the maximum size of the cache

* CCACHE_CPP2: This is to prevent certain pathalogical cases related to the preprocessor and enabling it is strongly recommended when using ccache with clang.

* CCACHE_HARDLINK: If true, ccache will attempt to use hard links from the cache directory when creating the compiler output rather than using a file copy. Using hard links may be slightly faster in some situations

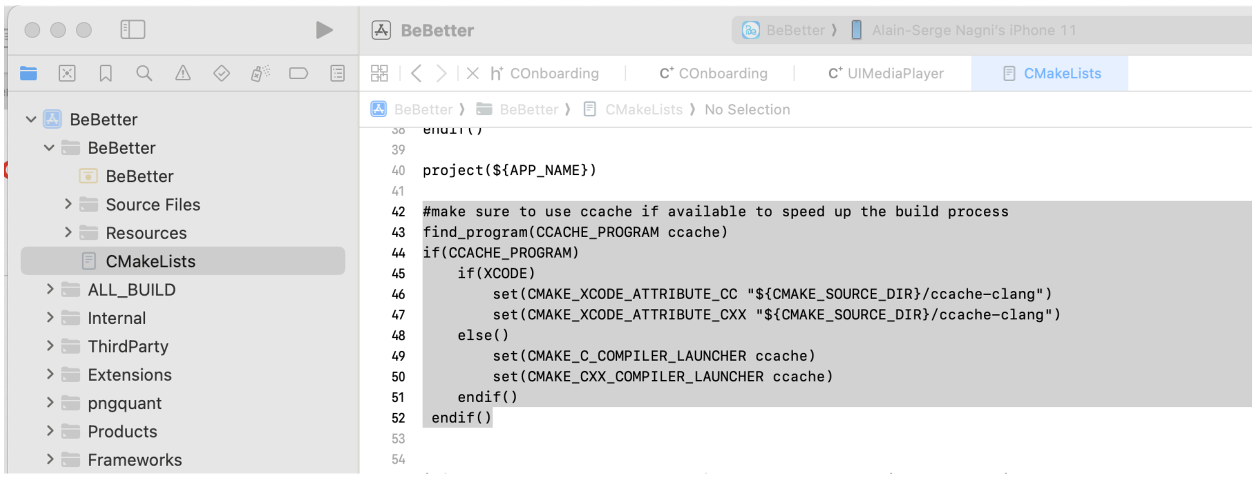
Step 3) Put the the ccache-clang (script) in project folder. Where the CMake file resides.



Step 4) change the access permissions of “ccache-clang” file by issuing the following command (from the project folder):

chmod 777 ccache-clang

Step 5) Modify the CMake file to use ccache. The change that I’m proposing will not attempt to use ccache if it’s not installed on your system. That makes it easier if you working in a team. It gives the time to your team members to adjust at their own speed.



```
endif()

project(${APP_NAME})

#make sure to use ccache if available to speed up the build process
find_program(CCACHE_PROGRAM ccache)
if(CCACHE_PROGRAM)
    if(XCODE)
        set(CMAKE_XCODE_ATTRIBUTE_CC "${CMAKE_SOURCE_DIR}/ccache-clang")
        set(CMAKE_XCODE_ATTRIBUTE_CXX "${CMAKE_SOURCE_DIR}/ccache-clang")
    else()
        set(CMAKE_C_COMPILER_LAUNCHER ccache)
        set(CMAKE_CXX_COMPILER_LAUNCHER ccache)
    endif()
 endif()
```

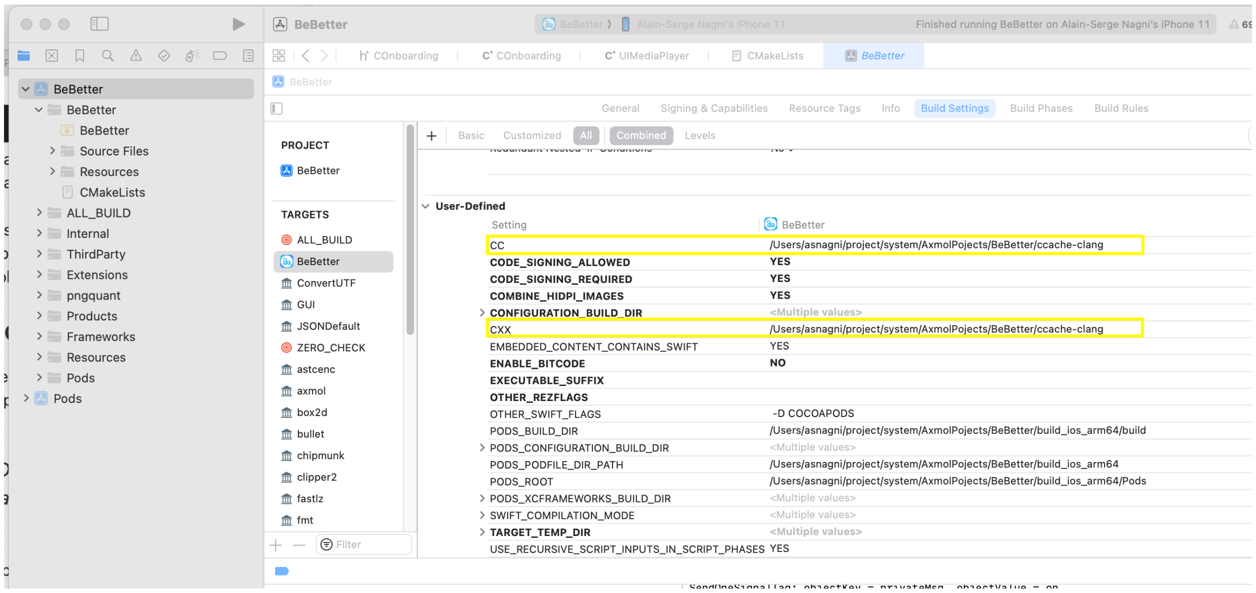
This is the CMake code to add to your CMake file:

```
find_program(CCACHE_PROGRAM ccache)
if(CCACHE_PROGRAM)
   if(XCODE)
      set(CMAKE_XCODE_ATTRIBUTE_CC "${CMAKE_SOURCE_DIR}/ccache-clang")
      set(CMAKE_XCODE_ATTRIBUTE_CXX "${CMAKE_SOURCE_DIR}/ccache-
clang")
   else()
      set(CMAKE_C_COMPILER_LAUNCHER ccache)
      set(CMAKE_CXX_COMPILER_LAUNCHER ccache)
   endif()
 endif()
```

<u>Step 5)</u> Add the two values "CC" & "CXX" into xcode Build Setting (User Defined Setting).

When Xcode projects contain user-defined build settings with the names `CC` and `CXX`, they override what Xcode uses as the compilers for C and C++ sources respectively. The compile command will be redirected to ccache by using the "CC" and "CXX" attributes.

Give these parameters the value: **${SOURCE_ROOT}/ccache-clang**



User-Defined

Setting	BeBetter
CC	/Users/asnagni/project/system/AxmolPojects/BeBetter/ccache-clang
CODE_SIGNING_ALLOWED	**YES**
CODE_SIGNING_REQUIRED	**YES**
COMBINE_HIDPI_IMAGES	**YES**
CONFIGURATION_BUILD_DIR	<Multiple values>
CXX	/Users/asnagni/project/system/AxmolPojects/BeBetter/ccache-clang
EMBEDDED_CONTENT_CONTAINS_SWIFT	YES
ENABLE_BITCODE	**NO**
EXECUTABLE_SUFFIX	
OTHER_REZFLAGS	
OTHER_SWIFT_FLAGS	-D COCOAPODS
PODS_BUILD_DIR	/Users/asnagni/project/system/AxmolPojects/BeBetter/build_ios_arm64/build
PODS_CONFIGURATION_BUILD_DIR	<Multiple values>
PODS_PODFILE_DIR_PATH	/Users/asnagni/project/system/AxmolPojects/BeBetter/build_ios_arm64
PODS_ROOT	/Users/asnagni/project/system/AxmolPojects/BeBetter/build_ios_arm64/Pods
PODS_XCFRAMEWORKS_BUILD_DIR	<Multiple values>
SWIFT_COMPILATION_MODE	<Multiple values>
TARGET_TEMP_DIR	<Multiple values>
USE_RECURSIVE_SCRIPT_INPUTS_IN_SCRIPT_PHASES	YES

<u>Step 5)</u> Compile again to build the ccache system. After that when you will build again it will be 8x faster.