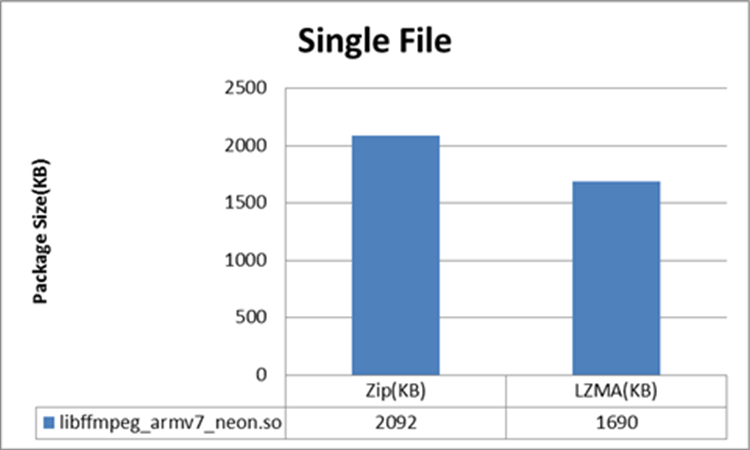
Android\* Apps Native Library Compression SDK

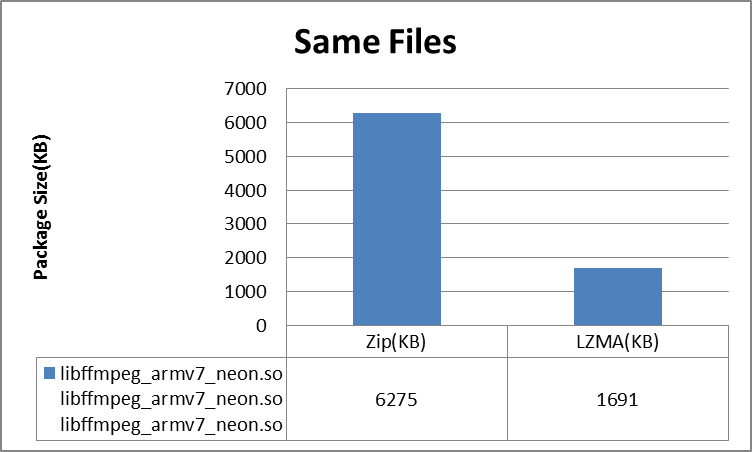
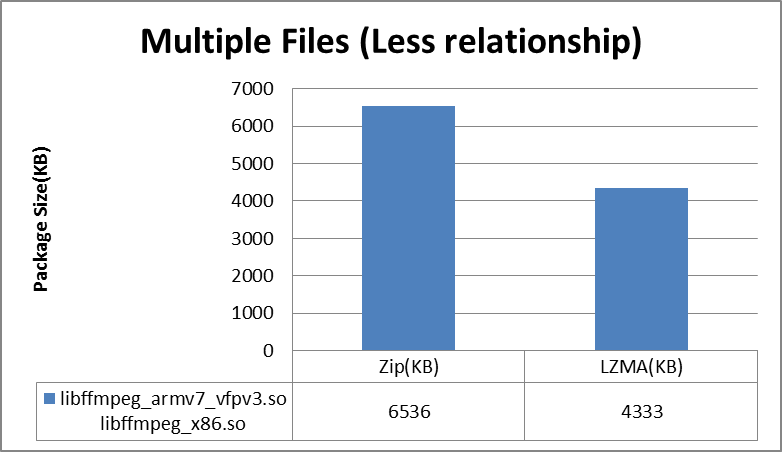
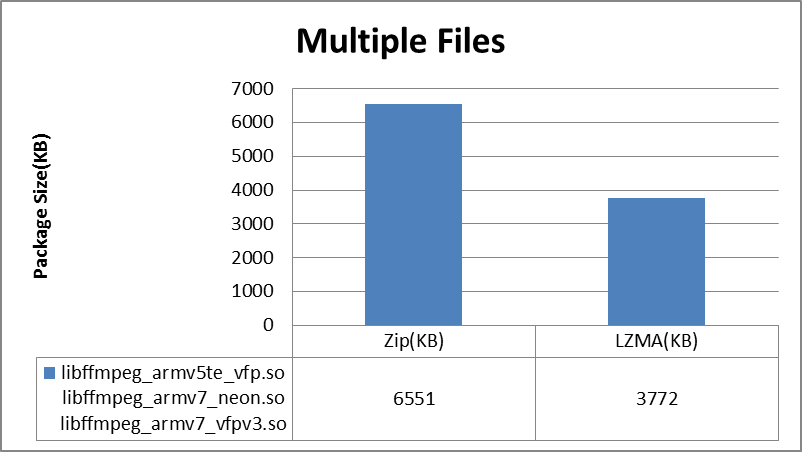
# *Introduction*

Android apps are typically written in Java, with its elegant object-oriented design and cross-platform feature. However, at times, you need to overcome the limitations of Java, such as memory management and performance, by programming directly into Android native interface, or just reusing the previous C/C++ code. Android provides Native Development Kit (NDK) to support native development in C/C++, besides the Android Software Development Kit (Android SDK) which supports Java.

There are lots of NDK apps on Google software market. Some NDK apps only contain arm library, which may cause low performance and high power consumption. Lots of reasons cause this issue, such as the package size, more time for test and even some business issues. Especially for package size, some ISV will release single apk while not fat apk (Here single apk means only containing arm library, while fat apk has both). There are some shortcomings for single apk. The single application may have delay update version. X86 single apk may be hard to be gotten by end-users. Application updating may overlap the x86 single apk with arm version. So how to encourage ISV making fat apk is a critical issue for x86 android software ecosystem.

# *Zip vs LZMA*

The author has developed a native library compression sdk to solve the apk size problem. The core idea is using LZMA (7z uses LZMA) to compress the native library. Google use zip to compress all of the content, zip is fast but the compress rate is lower than LZMA. Benefit from the large dictionary size, LZMA is especially suitable for native library compression. The following data will show you the comparison rate between Zip and LZMA.



From the four charts, we can get a conclusion that LZMA can reduce the redundancy between the files. In the extreme (same files), LZMA can get the highest compression rate according to Zip. This feature is particularly suitable for the compression of native library. Generally native library will use the same code to get “armeabi”,”armeabi-v7a”,”x86” or even “mips” library, and for “armeabi-v7a” there are neon and non-neon code too. These libraries have redundancy due to the same source code. And even for different architecture, for example, libffmpeg\_armv7\_vfpv3.so and libffmpeg\_x86.so, one is armeabi and the other is x86, the compression rate is 40%, while for single file, the rate is only 20%.

# *Native Library Compression SDK*

Native library compression sdk is developed by the author and it can make ISV to integrate LZMA native library compression and get smaller size. The core idea of this SDK is compress the entire native library into asserts folder. On the first run of this application, extract the entire native library from asserts folder.

The API in this SDK is very simple.

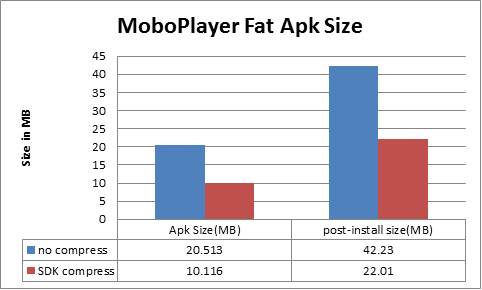
*static boolean NewInstance(Context cont,Handler hdl,boolean showProgress)*

*static DecRawso GetInstance()*

*String GetPath(String libname)*

ISV should modify the source code and only adds **NewInstance** when the application starts, and changes **system.loadlibrary(\*\*\*)** to **system.load(DecRawso . GetInstance ().GetPath(\*\*\*)).** After these Minor changes, the apk can get smaller size and run the same as before.

If ISV can make sure an enough delay from the calling of **NewInstance** to the first loading of native library, synchronous call (**NewInstance (cont,null,false)**) is recommended. Otherwise, a **Handler** should be passed to receive the “decode end” asynchronous message.

The author has work with Clover to integrate this SDK to MoboPlayer. They call **NewInstance** in synchronization methodwhen users enter the splash screen. The end-users can’t aware of it since all will be done on background service. The following chart shows the compress result.

# *Enhance functions of Native Library Compression SDK*

Besides the LZMA compression, this SDK will provide more functions to encourage ISV to pack x86 native library.

1. Cloud Storage: ISV can store the x86 libraries on the cloud server. And these libraries will be downloaded from the server after installation. This action is automatically done on x86 devices and only enable when WIFI is connected.
2. Missing Library Detection: If x86 libraries are missing, the SDK will automatically re-extract arm libraries. ISV can copy the arm library to the x86 folder to avoid this issue. But ISV must make sure there is no cross-reference between arm and x86 libraries.
3. Java Tool for Package: a java package tool is provided to convert normal apk to LZMA compress apk. This tool supports windows, Linux and Mac system. You can use it as: **ComPressApk.jar -a C:/my/test.apk -k c:/key \*\*\* ### alias name -x86** [**http://www.test.com**](http://www.test.com), if “-k” is missing, eclipse default test key will be used to sign this apk. The java tool can be integrated into ants build script to support automatically compile and publish.  
   (if name is null, the default name is CERT)
4. Filter: for ex: **ConfigureFilter("libffmpeg", "libffmpeg\_armv7\_neon.so"),** that means only extract libffmpeg\_armv7\_neon.so among all of the libraries that start with “libffmpeg”. That is useful to decrease the post-install size especially on arm devices.

# *Conclusion*

The Android Apps Native Library Compression SDK can cancel the worry of developers to deliver the fat apk. For the source and technique support, please contact the author.

**Notices**

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.  
  
UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.  
  
Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.   
  
The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.   
  
Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.   
  
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark\* and MobileMark\*, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

Copyright © 2014 Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.