An Analysis of Existing Android Image Loading Libraries:  
Picasso, Glide, Fresco, AUIL and Volley

Yoo-jeong SONG, Soo-bin OU and Jong-woo LEE*  
Dept. of Information Technology Engineering, Sookmyung Women’s University Cheongpa-ro 47-gil  
100, Youngsan-Ku, Seoul, 140-742, Korea  
*Corresponding author

Keywords: Image loading, Android, 3rd party library, Picasso, Glide, Fresco, AUIL, Volley.

Abstract. Recently, a large amount of development of Android applications being developed because of an increase in smartphone demand. Social network applications are one of the popular applications. Almost all these applications handle photos or images. Android application developers feel burden on processing images or photos in applications. For this reason, android third party libraries which refine inconvenience are developed. In this paper, we compare and analyze Picasso, Glide, Fresco and AUIL which are android third party libraries used for image processing. Furthermore, we also compare and analyze Volley framework. Our results can help image loading library users choose the appropriate one among them.

Introduction  
Android can help us to develop a variety of applications because it is based on open source platform. In 2016, Industry analysis and consulting firm IDC was expected to increase the demand for smart phones equipped with the Android operating system[1]. Accordingly, demand of android application using ratio is increasing. In Korea Google play store, Social network service applications hold a high rank, for example, Kakao Story, Facebook, Instagram. These applications mainly use photos or images for communicating other people. Photo or image is a simple and quick way to record the user's day-to-day or event. However, processing and displaying the photos or images in the application is quite complex. For solving this problem, there are many third party libraries that can process images using a simple code, for example, Picasso, Glide, Fresco, AUIL and Volley.

In this paper, we use these five image loading libraries to load and display the image. We compare and analyze these libraries and extract each library’s feature. As a result, we provide the information and guideline when programmer selects image library.

The rest of the paper proceeds as follows. In Related Work section, we discuss the related work. In Comparison Analysis Section, we describes the comparison analysis of five image loading libraries. In Conclusion section, we highlights conclusion.

Related Work

Picasso and Glide

Picasso is released from Square, which is composed of many contributors for the many open source projects[2]. It is optimized for downloading image or caching. It helps intuitive and convenient programming because it uses method chaining. Picasso provides the Snapshot function that is text about all image information. It also provides a unique function that represents the color of the tag to the image, where it came from. Picasso library help us to load an android resource image, file image and network image. Glide is released from Bumptech. It also uses method chaining like Picasso. Therefore it helps us to develop application intuitive and convenient. Glide supports fetching, decoding, and displaying video stills, images, and animated GIFs. Glide's primary focus is on making...
scrolling any kind of a list of images as smooth and fast as possible, but Glide is also effective for almost any case where you need to fetch, resize, and display a remote image.

Fresco

Fresco is image loading library that is provided by Facebook. It uses image pipeline method in image loading. Image pipeline can bring images from network, local storage and local resource. Fresco is possible to use even a low-end device, and it is also possible to automatically refresh the image data received by streaming. Also it supports GIFs as Glide and supports further WebP format[3].

AUIL

AUIL is an abbreviation of Android Universal Image Loader, and it is applied in many applications. This library uses cache policy which limits the capacity of the cache based on the screen size. It provides multi-thread image loading function, Bitmap options changing function and thread pool size modulating functions[4].

Volley

It proposed by Google in 2013, packaging of the Http asynchronous request API, providing an elegant and robust request queue, with better cache mechanism, allows network access to Android applications easier and more efficient[5]. Volley provides a NetworkImageView class that inherits the ImageView class. It not only provides network communications, but also provides automatically decoding function and image cache. Also, It can process image in batches when displaying many images on UI at a time. However, it uses a separate image view and image loader objects because its main feature is network communication. This library is difficult to use than others. However, there is an advantage to provide an efficient image downloads.

Comparison Analysis of Picasso, Glide, Fresco, AUIL and Volley

In this section, we describe the comparison analysis of Picasso, Glide, Fresco, AUIL and Volley. It has four steps. In the first step, we compare usage of them, such as method and algorithms. In the second step, we compare display image format and features. In the third step, we compare using memory ratio on mobile device. In the forth step, we make a comparison table overall comparison analysis result. In the fifth step, we explain the experiment environments.

Usage of Picasso, Glide, Fresco, AUIL and Volley. In this section, we explain the usage of Picasso, Glide, Fresco, AUIL and Volley. Basically, Picasso and Glide’s usage is very similar to each other. These are using method changing method. Many functions have same form and usage, but some are different. Method chaining is a common syntax for invoking multiple method calls in object-oriented programming languages. Each method returns an object, allowing the calls to be chained together in a single statement without requiring variables to save the intermediates results [6]. Fresco uses a custom view that called SimpleDraweeView, not typical image view being used. Before starting application, internal android system generate view. Fresco should be initialized before generating this view. Then, we can set URI in SimpleDraweeView. Fresco can allow us to load image easily, but it only uses custom image view supporting by Fresco. For using AUIL, we set image display options and make image loader. We create DisplayImageOptions object and ImageLoader object. DisplayImageOptions object can set options such as disk cache and memory cache. Volley is not image loading library. So, it is not easy to use unlike other libraries. It is a framework that enables restful communication. However, this framework support image loading function on networking system. In order to load images efficiently, we create NetworkImageView object. We add the request to request queues for loading the image. And it passes a callback object for processing of the results.

Display Image Format and Image Cache. In this section, we explain the basic image of each library’s display image format and image cache. Picasso uses ARGB_8888 bitmap format essentially, but Glide uses RGB_565 bitmap format. Therefore, image loaded by Picasso has more quality than image loaded by Glide[2]. In case of Picasso, images are processed the original image when image
caching. For example, 1920x1080 image is cached 1920x1080. Glide caches as much as the size of the image view. For example, 1920x1080 image is cached 384x216 when the size of image view is 384x216. Fresco uses ARGB_8888 bitmap format. However, Fresco uses special memory region called ashmem and saves image in there. In general, it does not account for the Java heap memory. So, It reduces the probability Out Of Memory occurs. AU1L also uses the ARGB_8888 image format. We can change this format according to the user’s needs. AU1L uses LRU cache policy. LRUcache releases the memory that is not used the most recently. We can use the other cache policy needed. Volley uses RGB_565 image format. Volley converts original image size to image view size when displaying image. Volley image cache can be supported by Volley-caches that is able to use AU1L library cache intactly.

**Comparison of Memory Possession.** In this section, we explain the memory possession of Picasso, Glide, Fresco, AU1L and Volley when loading same image file. Fig. 1 is a graph about memory possession when loading image using five image loading libraries. We can find this result at memory monitor in android studio. According to graph, Picasso uses 59.40MB memory, and Glide uses 29.78MB memory. Although it is loading the same image file, the memory possession is quite different because of the different image processing between Picasso and Glide. Picasso brings a real image on memory first, and then resizes in GPU. But Glide brings image which is already resized on memory[2]. Fresco uses 49.35MB memory. It is using same image format of Picasso, however, it seems to take up less memory than Picasso because of using a different memory region. AU1L uses 49.67MB memory. It is less than Picasso, and is similar Fresco memory possession. Volley uses 33.84MB memory. It’s possession is more than Glide, but less than other libraries.

![Memory Possession Graph](image)

**Figure 1.** The memory possession graph.

**Compare and Analyze Results.** In this section, we explain the result of comparison analysis. Picasso can use easy and simple code for loading image because of method chaining. Because its basic image format is ARGB_8888, we can loading high quality image. However, OutOfMemoryError must be well managed to avoid because of high memory usage in the device according to the quality. Glide also uses method chaining and can load image more faster than Picasso because of image loading process. Fresco uses custom image view and special memory ashmem that is main feature of Fresco. AU1L had been used mainly before appearing Picasso or Glide. Usage is a little difficult, but easier than Volley. Advantage of this library is using of selective cache policy. Volley has a difficult
usage. However, it is able to load image more faster on network communication environment. And it also provide parallel image loading function. Table 1. is table about result of comparison analysis.

Table 1. The result of comparison analysis.

<table>
<thead>
<tr>
<th>Default Image format</th>
<th>Picasso</th>
<th>Glide</th>
<th>Fresco</th>
<th>AUIL</th>
<th>Volley</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image format</td>
<td>ARGB_8888</td>
<td>RGB_565</td>
<td>ARGB_8888</td>
<td>ARGB_8888</td>
<td>RGB_565</td>
</tr>
<tr>
<td>Memory Possession</td>
<td>59.40MB</td>
<td>29.78MB</td>
<td>49.35MB</td>
<td>49.67MB</td>
<td>33.84MB</td>
</tr>
<tr>
<td>Features</td>
<td>· Using Method chaining</td>
<td>· Using Method chaining</td>
<td>· Using custom view</td>
<td>· Various cache policies available</td>
<td>· Network Communication Framework · Difficult usage · Supporting Volley Extension library</td>
</tr>
</tbody>
</table>

Experiments Environment. In this section, we explain experimental environment about image loading experiments on smartphone using five image loading libraries. For the experiments in this paper, Android Studio are used for developing android applications. We used the LG G4 that is based to the version of android 5.1. The images used in the experiment are uploaded to a web page and we can receive this image comes through the URI.

Conclusion

In this paper, we compare and analysis Picasso, Glide, Fresco, AUIL and Volley among android third party library. Through this study, we provide guidelines that can be used selectively in developing image or photos—related applications for android programmer. For this comparison analysis and experiment, we find out about the basic image format and image cache policy. Also, we find out about five libraries’ difference of memory possession. We should be careful when developing because memory management is very important issues in android application developing.

In the future researches, we measure displaying time when using each of five libraries. Using the measurement result, we also propose an application implementation about big data image of high quality image processing using the most suitable and fastest library.

Acknowledgement

This work was supported by National Research Foundation of Korea, Korea (2013R1A1A2013155, Development of a False Recognition Tolerable POI Voice Search Technique).

References
