Національний технічний університет України
“Київський політехнічний інститут ім І. Сікорського”

Звіт

з лабораторної роботи №6
на тему “Робота з кешем”

Виконав: студент групи
ІС-42 Коростильов Євгеній

Київ 2017 р

Кеш в Android

Для початку звернемось до Вікіпедії за означенням кешу.

Кеш (англ. Cache, від фр. Cacher - «ховати», вимовляється [kæʃ] - «кеш») - проміжний буфер з швидким доступом, що містить інформацію, яка може бути запрошена з найбільшою ймовірністю. Доступ до даних в кеші здійснюється швидше, ніж вибірка вихідних даних з більш повільної пам'яті або віддаленого джерела, однак її обсяг істотно обмежений у порівнянні зі сховищем вихідних даних.

Стає зрозуміло для чого ця технологія в мобільних додатках. Звернемось до офіційної документації за більш детальними даними.

Кеш пам'яті корисний для прискорення доступу до нещодавно переглянутих растрових образів, однак, не можна покладатися на зображення, доступні в цьому кеш-пам'яті. Компоненти, такі як GridView з великими наборами даних, можуть легко заповнити кеш пам'яті. Заявка може бути перервана іншим завданням, як телефонний дзвінок, і в той час як у фоновому режимі це може бути вбито, а кеш пам'яті знищено. Після того, як користувач відновить, заявка повинна знову обробляти кожне зображення.

Кеш-пам'ять диску може бути використана в таких випадках, щоб зберігати оброблені растрові зображення та зменшити час завантаження, коли зображення більше не доступні в кеші пам'яті. Звичайно, завантаження зображень із диска відбувається повільніше, ніж завантаження з пам'яті, і це слід зробити у фоновому потоці, оскільки час читання дисків може бути непередбачуваним.

Зразок коду цього класу використовує реалізацію DiskLruCache, яка витягується з джерела Android. Ось оновлений код прикладу, який крім пам'яті кешу пам'яті додає кеш-пам'ять:

```
private DiskLruCache mDiskLruCache;
private final Object mDiskCacheLock = new Object();
private boolean mDiskCacheStarting = true;
private static final int DISK_CACHE_SIZE = 1024 * 1024 * 10; // 10MB
```

```
private static final String DISK_CACHE_SUBDIR = "thumbnails";

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    // Initialize memory cache
    ...
    // Initialize disk cache on background thread
    File cacheDir = getDiskCacheDir(this, DISK_CACHE_SUBDIR);
    new InitDiskCacheTask().execute(cacheDir);
    ...
}

class InitDiskCacheTask extends AsyncTask<File, Void, Void> {
    @Override
    protected Void doInBackground(File... params) {
        synchronized (mDiskCacheLock) {
            File cacheDir = params[0];
            mDiskLruCache = DiskLruCache.open(cacheDir, DISK_CACHE_SIZE);
            mDiskCacheStarting = false; // Finished initialization
            mDiskCacheLock.notifyAll(); // Wake any waiting threads
        }
        return null;
    }
}

class BitmapWorkerTask extends AsyncTask<Integer, Void, Bitmap> {
    ...
    // Decode image in background.
    @Override
    protected Bitmap doInBackground(Integer... params) {
        final String imageKey = String.valueOf(params[0]);
```

```
        // Check disk cache in background thread
        Bitmap bitmap = getBitmapFromDiskCache(imageKey);

        if (bitmap == null) { // Not found in disk cache
            // Process as normal
            final Bitmap bitmap = decodeSampledBitmapFromResource(
                    getResources(), params[0], 100, 100));
        }

        // Add final bitmap to caches
        addBitmapToCache(imageKey, bitmap);

        return bitmap;
    }
    ...
}

public void addBitmapToCache(String key, Bitmap bitmap) {
    // Add to memory cache as before
    if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }

    // Also add to disk cache
    synchronized (mDiskCacheLock) {
        if (mDiskLruCache != null && mDiskLruCache.get(key) == null) {
            mDiskLruCache.put(key, bitmap);
        }
    }
}
```

```
public Bitmap getBitmapFromDiskCache(String key) {
    synchronized (mDiskCacheLock) {
        // Wait while disk cache is started from background thread
        while (mDiskCacheStarting) {
            try {
                mDiskCacheLock.wait();
            } catch (InterruptedException e) {}
        }
        if (mDiskLruCache != null) {
            return mDiskLruCache.get(key);
        }
    }
    return null;
}

// Creates a unique subdirectory of the designated app cache directory. Tries to
use external
// but if not mounted, falls back on internal storage.
public static File getDiskCacheDir(Context context, String uniqueName) {
    // Check if media is mounted or storage is built-in, if so, try and use
external cache dir
    // otherwise use internal cache dir
    final String cachePath =

Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState()) ||
                    !isExternalStorageRemovable() ?
getExternalCacheDir(context).getPath() :
                            context.getCacheDir().getPath();

    return new File(cachePath + File.separator + uniqueName);
}
```

В цьому прикладі ми бачимо як слід використовувати кеш для Бітмапів. Це дозволяє значно зменшити лаги і відповідно значно підвищити продуктивність.

Висновок

В цій лабораторній роботі я навчився використовувати кеш в Андроід додатках. Це дозволило ознайомитися з простішою технологію оптимізації додатків навіть на старих пристроях.