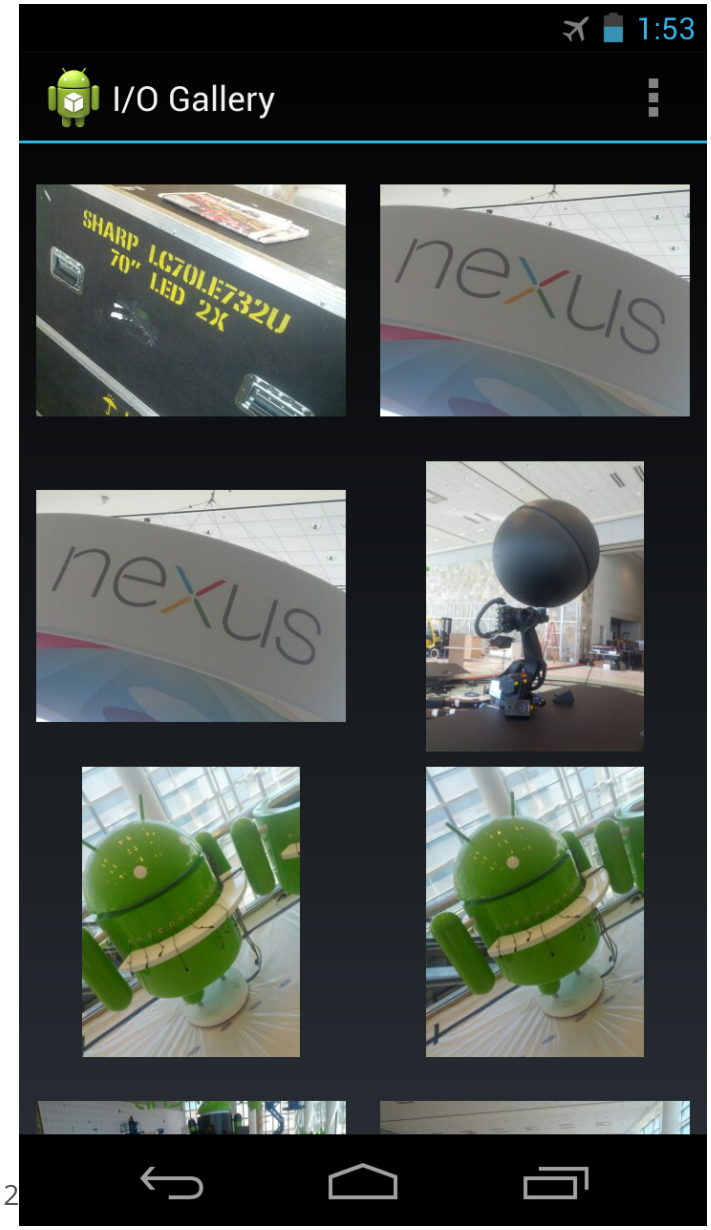

Google
Developers

Doing More With Less

Being a Good Android Citizen

Jeff Sharkey
Software Engineer, Android

What we want to build



How we're going to build it

- `MediaStore.Images` ContentProvider surfaces photos on device
  - `MediaStore.Images.Media.EXTERNAL_CONTENT_URI` for list of all photos
    - Each photo has a `BaseColumns._ID`
  - `MediaStore.Images.Thumbnails.getThumbnail()` to load a thumbnail
  - `ContentResolver.openInputStream()` to load a full resolution photo
- Walking through code snippets today, open source link at end

List adapter

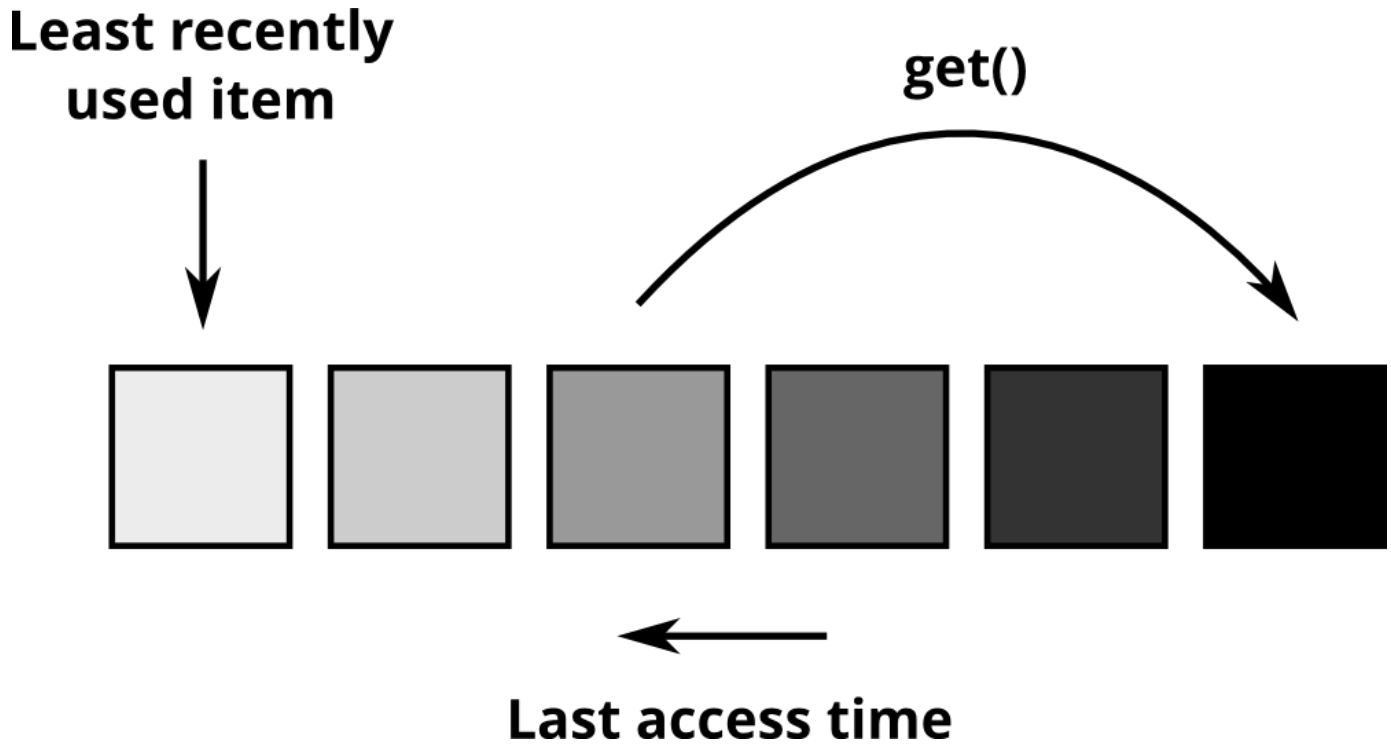
API 3

```
private class PhotoAdapter extends CursorAdapter {
    ...
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return new ImageView(context);
    }

    public void bindView(View view, Context context, Cursor cursor) {
        ImageView imageView = (ImageView) view;
        long photoId = cursor.getLong(cursor.getColumnIndex(BaseColumns._ID));

        new ThumbnailAsyncTask(imageView).execute(photoId);
    }
}
```

Async thumbnails

API 3

```
public class ThumbnailAsyncTask extends AsyncTask<Long, Void, Bitmap> {
    private ImageView mTarget;
    public ThumbnailAsyncTask(ImageView target) {
        mTarget = target;
    }
    protected Bitmap doInBackground(Long... params) {
        long photoId = params[0];
        return MediaStore.Images.Thumbnails.getThumbnail(
                getContentResolver(), photoId, MediaStore.Images.Thumbnails.MINI_KIND, null);
    }
    protected void onPostExecute(Bitmap result) {
        mTarget.setImageBitmap(result);
    }
}
```


Least recently
used item
get()
Last access time

Caching

- `LruCache` available in API 12, and in `support-v4` library

API 12

```
public class ThumbnailCache extends LruCache<Long, Bitmap> {
    public ThumbnailCache(int maxSizeBytes) {
        super(maxSizeBytes);
    }
    @Override
    protected int sizeOf(Long key, Bitmap value) {
        // Return size of in-memory Bitmap, counted against maxSizeBytes
        return value.getByteCount();
    }
    public Bitmap put(Long key, Bitmap value);
    public Bitmap get(Long key);
}
```

- Uncompressed 320×240 `ARGB_8888` `Bitmap` is 307kB

Cache size

- Picking a cache size; too small vs. too large

```
ActivityManager am = (ActivityManager) getSystemService(Context.ACTIVITY_SERVIC
```

API 5

```
int memoryClassBytes = am.getMemoryClass() * 1024 * 1024;
ThumbnailCache cache = new ThumbnailCache(memoryClassBytes / 8);
```

- `getMemoryClass()` indicates how hard of a memory limit we should impose on ourselves to let system work best
- Typical values are 48MB on Nexus S, 64MB on Galaxy Nexus

Caching in task

API 3

```
public class ThumbnailAsyncTask extends AsyncTask<Long, Void, Bitmap> {
    ...
    protected Bitmap doInBackground(Long... params) {
        long photoId = params[0];
        Bitmap bitmap = MediaStore.Images.Thumbnails.getThumbnail(
                getContentResolver(), photoId, MediaStore.Images.Thumbnails.MINI_KIND, null);

        // Store loaded thumbnail in cache
        mCache.put(photoId, bitmap);

        return bitmap;
    }
    ...
}
```

Caching in adapter

API 3

```
private class PhotoAdapter extends CursorAdapter {
    ...
    public void bindView(View view, Context context, Cursor cursor) {
        ImageView imageView = (ImageView) view;
        long photoId = cursor.getLong(0);

        // Try looking for cache hit first
        Bitmap cachedResult = mCache.get(photoId);
        if (cachedResult != null) {
            imageView.setImageBitmap(cachedResult);
        } else {
            new ThumbnailAsyncTask(imageView).execute(photoId);
        }
    }
}
```

Trim callbacks

API 1

```
public interface ComponentCallbacks {
    // Called after all background apps have been killed
    void onLowMemory();
}
```

API 14

```
public interface ComponentCallbacks2 extends ComponentCallbacks {
    // System would like us to reduce our memory usage
    void onTrimMemory(int level);
}
```

- Higher trim levels mean we should be more aggressive about releasing
- Ready to `@Override` on `Activity`, `Fragment`, `Service`, `ContentProvider`, and `Application`
- When outside a specific lifecycle, you can listen using `Context.registerComponentCallbacks()`

Trim callbacks

- Well-defined trim values: `COMPLETE`, `MODERATE`, `BACKGROUND`, `UI_HIDDEN`

API 14

```
public void onTrimMemory(int level) {
    super.onTrimMemory(level);
    if (level >= TRIM_MEMORY_MODERATE) { // 60
        // Nearing middle of list of cached background apps; evict our
        // entire thumbnail cache
        mCache.evictAll();
    } else if (level >= TRIM_MEMORY_BACKGROUND) { // 40
        // Entering list of cached background apps; evict oldest half of our
        // thumbnail cache
        mCache.trimToSize(mCache.size() / 2);
    }
}
```

- Callbacks are purely advisory, but if we release memory, system may keep us cached longer

Starring photos

- Enter an action mode when photo is long-pressed

API 11

```
mGridView.setChoiceMode(CHOICE_MODE_MULTIPLE_MODAL);
mGridView.setMultiChoiceModeListener(new MultiChoiceModeListener() {
    ...
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        SparseBooleanArray checked = mGridView.getCheckedItemPositions();
        ...
    }
});
```

- Checked items are `int` → `boolean`
- Could store as `HashMap<Integer, Boolean>`, but instead it returns `SparseBooleanArray`

Autoboxing

- The difference between `int` and `java.lang.Integer`

API 1

```
// Allocates 0 objects
int total = 0;
for (int i = 0; i < 100; i++) {
    total += i;
}
```

API 1

```
// Allocates 86 new objects
Integer total = 0;
for (int i = 0; i < 100; i++) {
    total += i;
}
```

Avoiding autoboxing

- Overhead of storing 100 key/value pairs:

`java.util`	`android.util`	`java.util` **objects**	`android.util` **objects**
`HashMap<Integer, Boolean>`	`SparseBooleanArray`	202	3
`HashMap<Integer, Integer>`	`SparseIntArray`	302	3
`HashMap<Integer, E>`	`SparseArray<E>`	302	103
`HashMap<Long, E>`	`LongSparseArray<E>`	302	103

SQLite

- Changes are ACID (atomic, consistent, isolated, durable)

API 1

```
SQLiteDatabase db;
long[] selectedIds;

ContentValues values = new ContentValues();
for (int i = 0; i < selectedIds.length; i++) {
    values.put(COLUMN_ID, selectedIds[i]);
    values.put(COLUMN_STARRED, starred);

    db.insert(TABLE_STARRED, null, values);
}
```

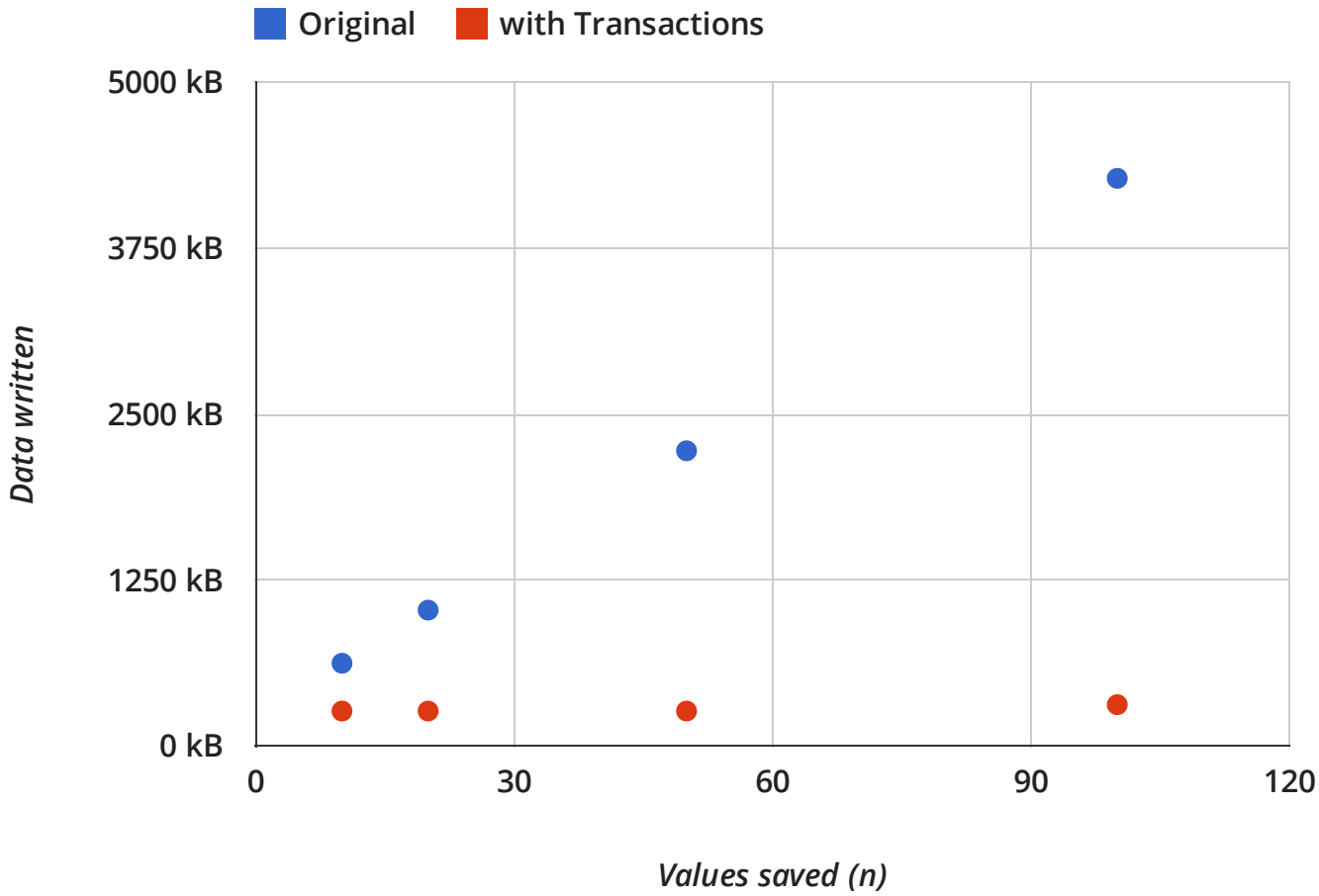
- Each `insert()` has overhead, which can add up quickly

SQLite transactions

- Better performance if we wrap in a transaction:

API 1

```
db.beginTransaction();
try {
    for (int i = 0; i < selectedIds.length; i++) {
        values.put(COLUMN_ID, selectedIds[i]);
        values.put(COLUMN_STARRED, starred);

        db.insert(TABLE_STARRED, null, values);
    }

    db.setTransactionSuccessful();
} finally {
    db.endTransaction();
}
```


Original
with Transactions
5000 kB
3750 kB
2500 kB
1250 kB
0 kB
Data written
0
30
60
90
120
Values saved (n)

Yielding

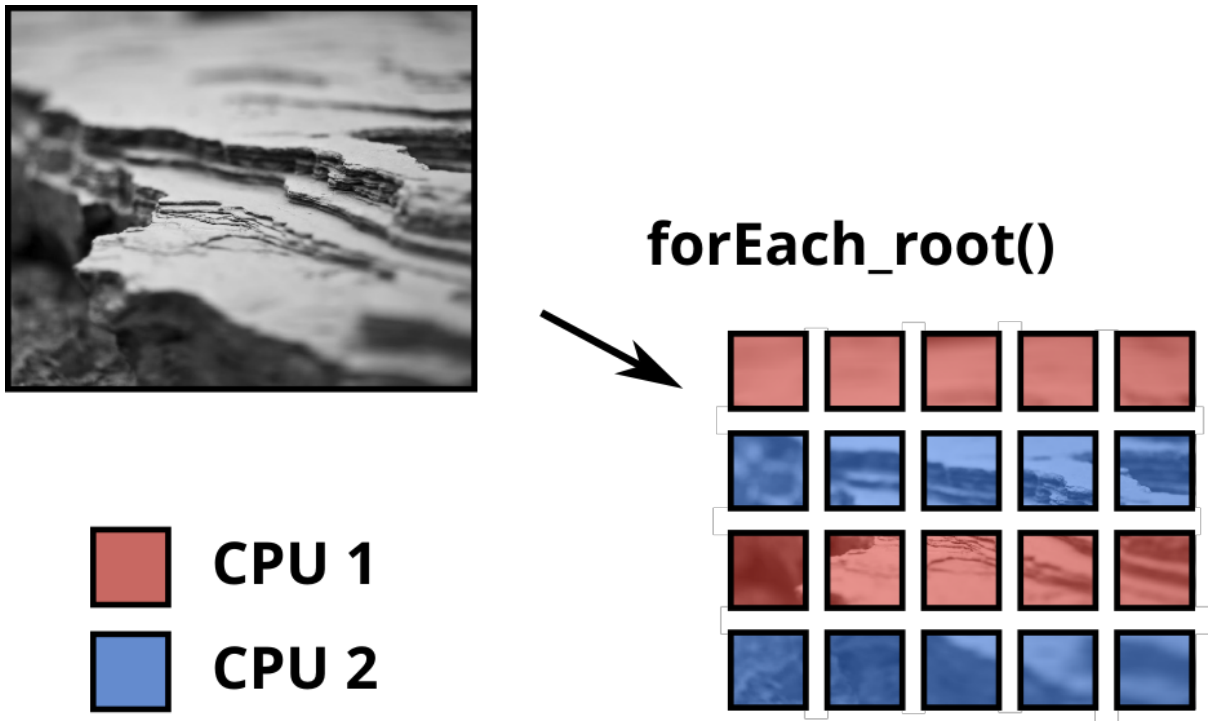
- By default, transactions hold an exclusive lock on database

API 3

```
db.beginTransaction();
try {
    for (int i = 0; i < selectedIds.length; i++) {
        values.put(COLUMN_ID, selectedIds[i]);
        values.put(COLUMN_STARRED, starred);

        db.insert(TABLE_STARRED, null, values);
        db.yieldIfContendedSafely();
    }

    db.setTransactionSuccessful();
} finally {
    db.endTransaction();
}
```

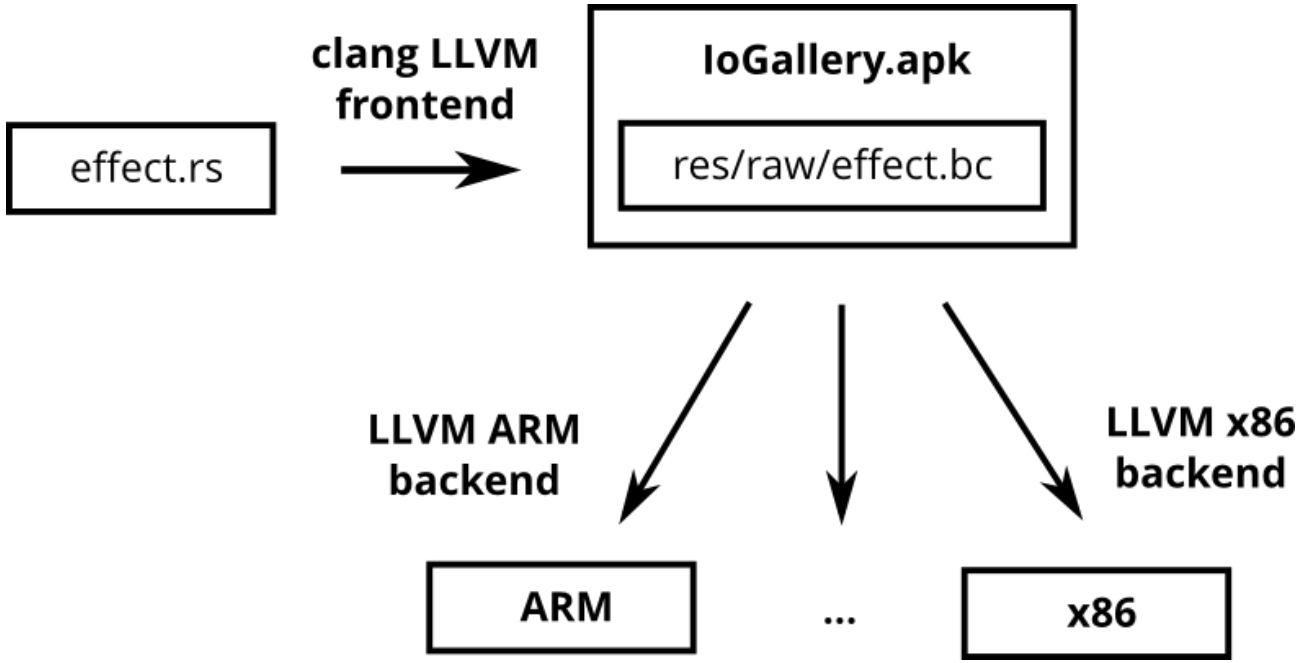
Scales to hardware

- Parallelizes by creating worker thread for each CPU core

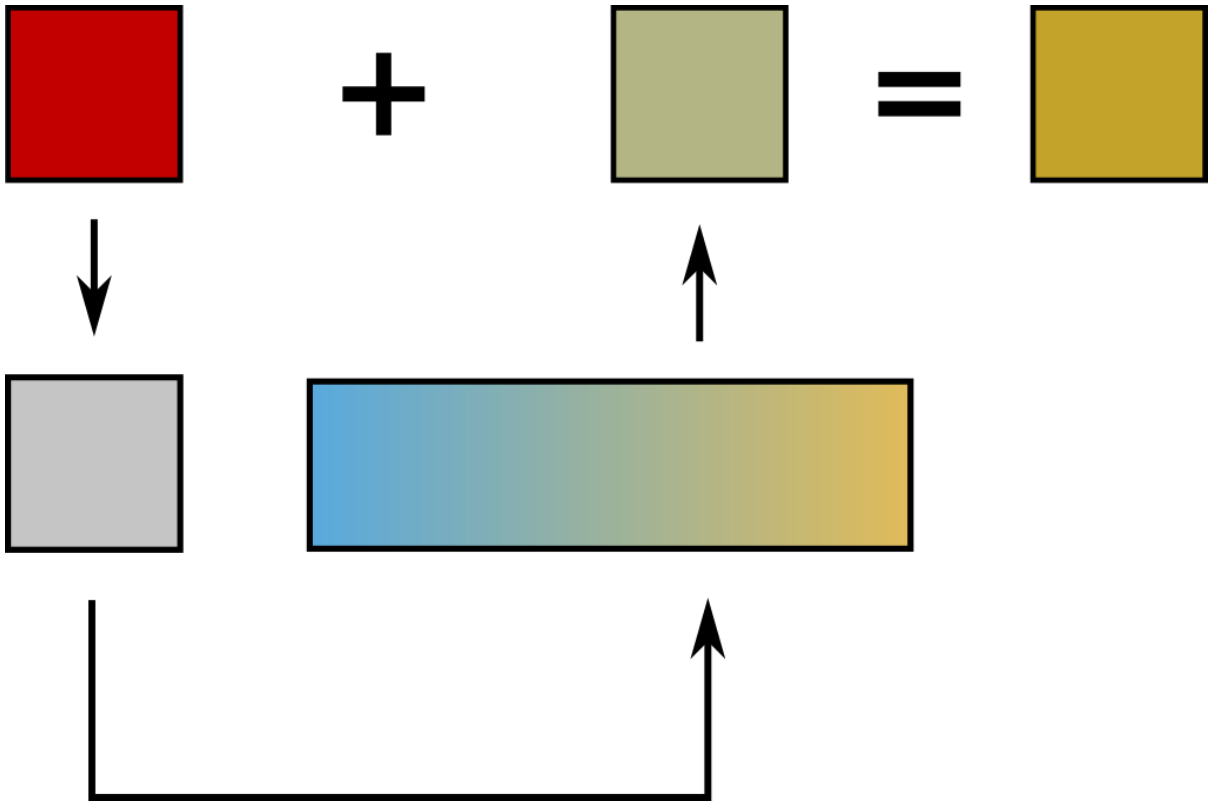


All architectures

- Renderscript compiles to intermediate bitcode
- Generates JNI glue code for easy access from Dalvik



Color filter



Renderscript

EFFECT.RS

```
#pragma version(1)
#pragma rs java_package_name(org.example.android.fastgallery)
#pragma rs_fp_relaxed(relaxed)

const static float4 convertIntensity = { 0.299f, 0.587f, 0.114f, 0.f };

void root(const uchar4 *inPixel, uchar4 *outPixel) {
    float4 inColor = convert_float4(*inPixel);

    // Compute intensity of current pixel
    float intensity = dot(convertIntensity, inColor) / 255.f;
    ...
```

- Vector types: `float4` is just four `float` packed into a single type

Renderscript

EFFECT.RS

```
...
float4 darkColor;
float4 lightColor;
float strength;

void root(const uchar4 *inPixel, uchar4 *outPixel) {
    ...
    // Mix between two color tones based on intensity
    float4 tonedColor = mix(darkColor, lightColor, intensity);
    tonedColor = tonedColor * intensity;

    // Blend toned and original colors
    float4 outColor = mix(inColor, tonedColor, strength);
    *outPixel = convert_uchar4(outColor);
}
```

Renderscript and Java

PHOTOACTIVITY.JAVA

```
RenderScript rs = RenderScript.create(this);

// Load photo into Dalvik-managed objects
Bitmap in = loadBitmap(getContentResolver(), getIntent().getData());
Bitmap out = Bitmap.createBitmap(in.getWidth(), in.getHeight(), in.getConfig());

// Copy photo into Renderscript-managed objects
Allocation inAlloc = Allocation.createFromBitmap(
        rs, in, MipmapControl.MIPMAP_NONE, Allocation.USAGE_SCRIPT);
Allocation outAlloc = Allocation.createTyped(
        rs, inAlloc.getType(), Allocation.USAGE_SCRIPT);
```

Renderscript and Java

PHOTOACTIVITY.JAVA

```
ScriptC_effect script = new ScriptC_effect(rs, getResources(), R.

// Bind script parameters
script.set_darkColor(convertHsvToFloat4(darkHue, .5f, .75f));
script.set_lightColor(convertHsvToFloat4(lightHue, .5f, .75f));
script.set_strength(strength);

// Run script across input bitmap
script.forEach_root(inAlloc, outAlloc);

// Copy result back to Dalvik, and show user
outAlloc.copyTo(out);
imageView.setImageBitmap(out);
```

Renderscript

- Time to process a 5 megapixel image:

Device	CPU	Time (ms)
Nexus S	1GHz ARM	1232
Galaxy Nexus	1.2GHz dual-core ARM	416
Nexus 7	1.2GHz quad-core ARM	243

- No code changes required to scale to hardware
- Targets all architectures, including SIMD optimizations
- No access to Bionic libc; not designed to replace NDK. Renderscript is best suited for compute intensive code.

Filter new photos

API 14

```
<receiver android:name=".PhotoReceiver">
    <intent-filter>
        <action android:name="android.hardware.action.NEW_PICTURE" />
        <data android:mimeType="image/*" />
    </intent-filter>
</receiver>
```

API 3

```
public class PhotoReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        if (isAutoApplyEnabled()) {
            // Perform work in IntentService to avoid ANR
            Intent serviceIntent = new Intent(context, EffectService.class);
            serviceIntent.setData(intent.getData());
            context.startService(serviceIntent);
        }
    }
}
```

Only when requested

API 14

```
<receiver android:name=".PhotoReceiver" android:enabled="false">
    <intent-filter>
        <action android:name="android.hardware.action.NEW_PICTURE" />
        <data android:mimeType="image/*" />
    </intent-filter>
</receiver>
```

API 1

```
PackageManager pm = context.getPackageManager();
ComponentName component = new ComponentName(context, PhotoReceiver.class);

int state = isAutoApplyEnabled()
        ? PackageManager.COMPONENT_ENABLED_STATE_ENABLED
        : PackageManager.COMPONENT_ENABLED_STATE_DEFAULT;
pm.setComponentEnabledSetting(component, state, PackageManager.DONT_KILL_APP);
```

- Important for common broadcasts like `CONNECTIVITY_CHANGE` and `PACKAGE_ADDED`

Summary

Doing More With Less

Open source code

http://code.google.com/p/iogallery/


Google
Developers