Acelera tus aplicaciones con Renderscript

Alejandro Acosta

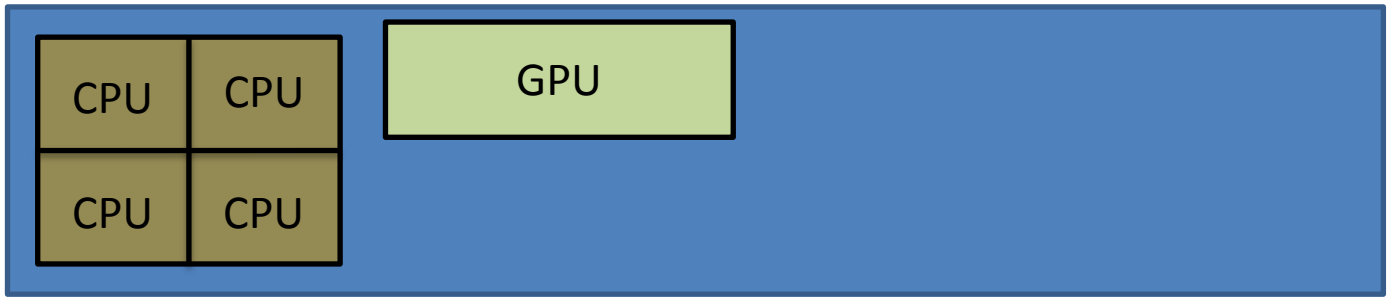
Índice

- Introducción
- Renderscript
  - Contexto
  - Memoria
  - Kernel
- Modelo de ejecución
- Ejemplo
- Intrinsics

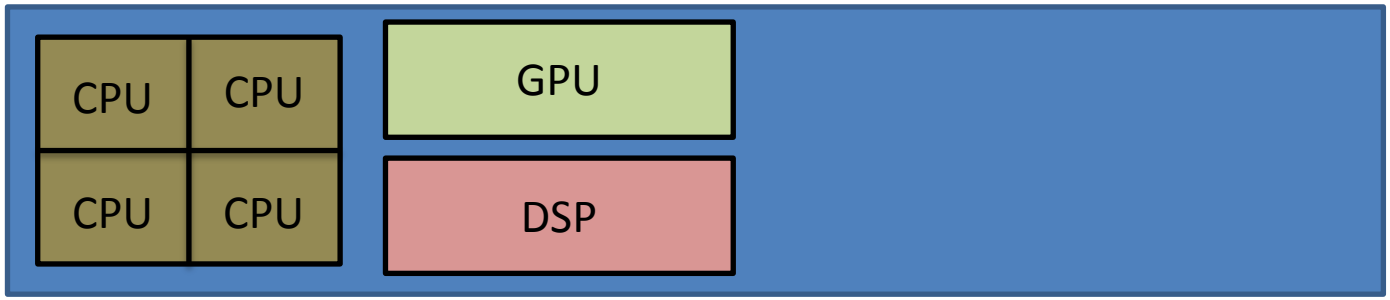
https://www.ifixit.com/Teardown/Nexus+7+Teardown/9623

Hardware

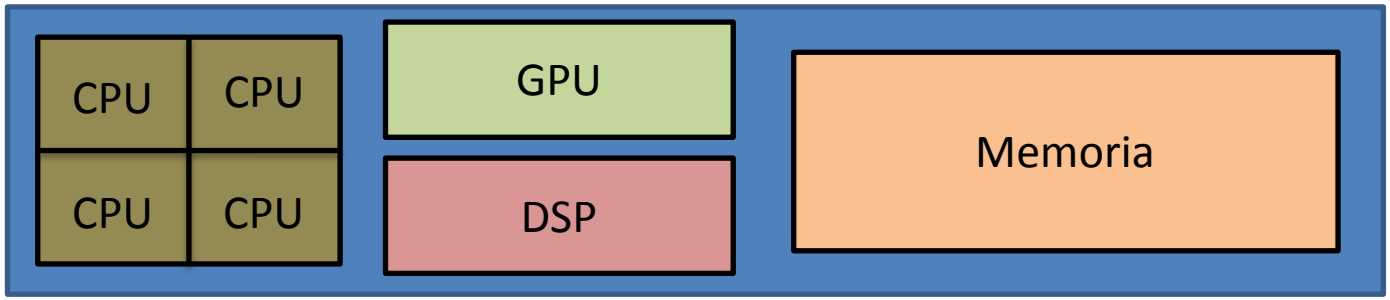
Hardware



Hardware

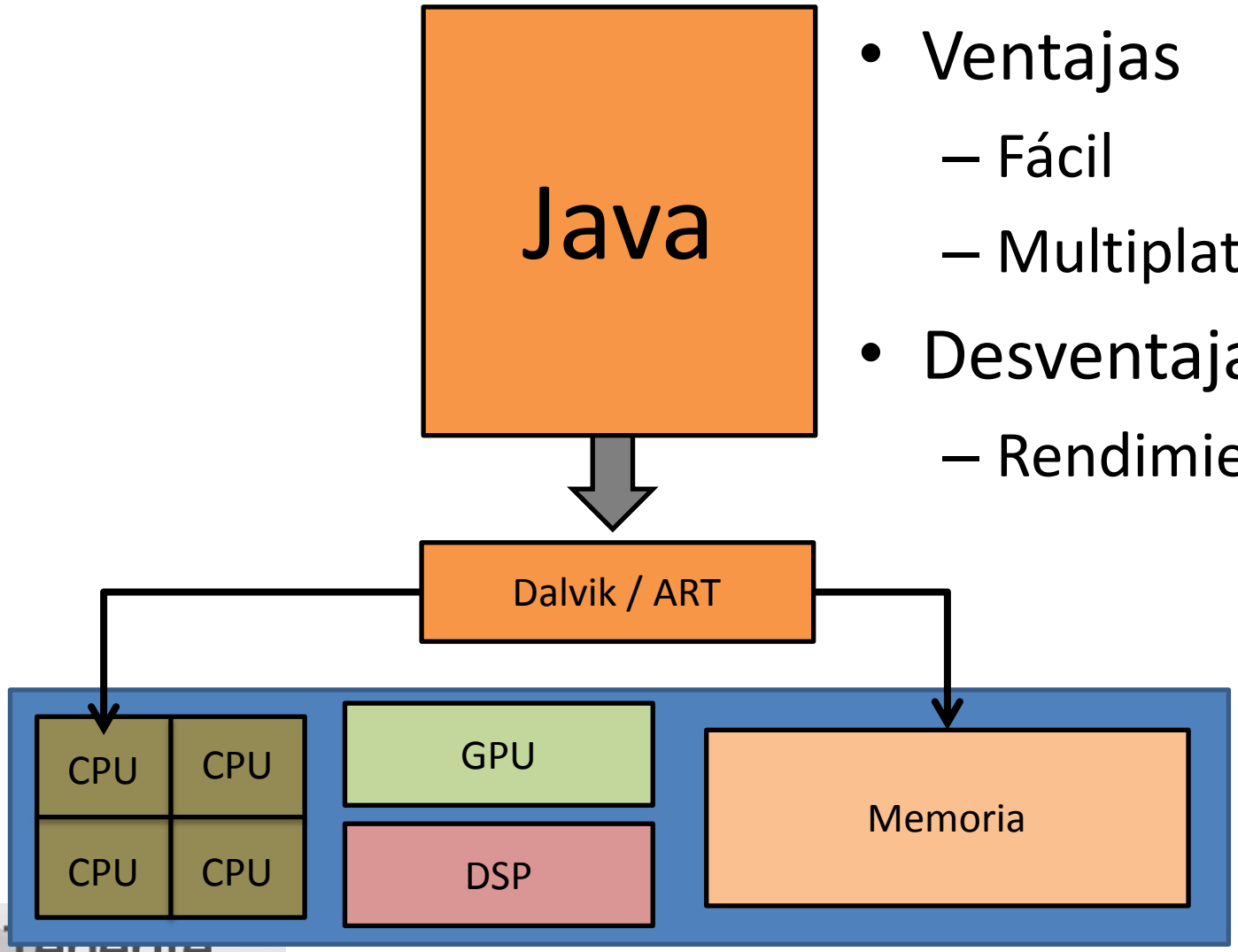


Hardware



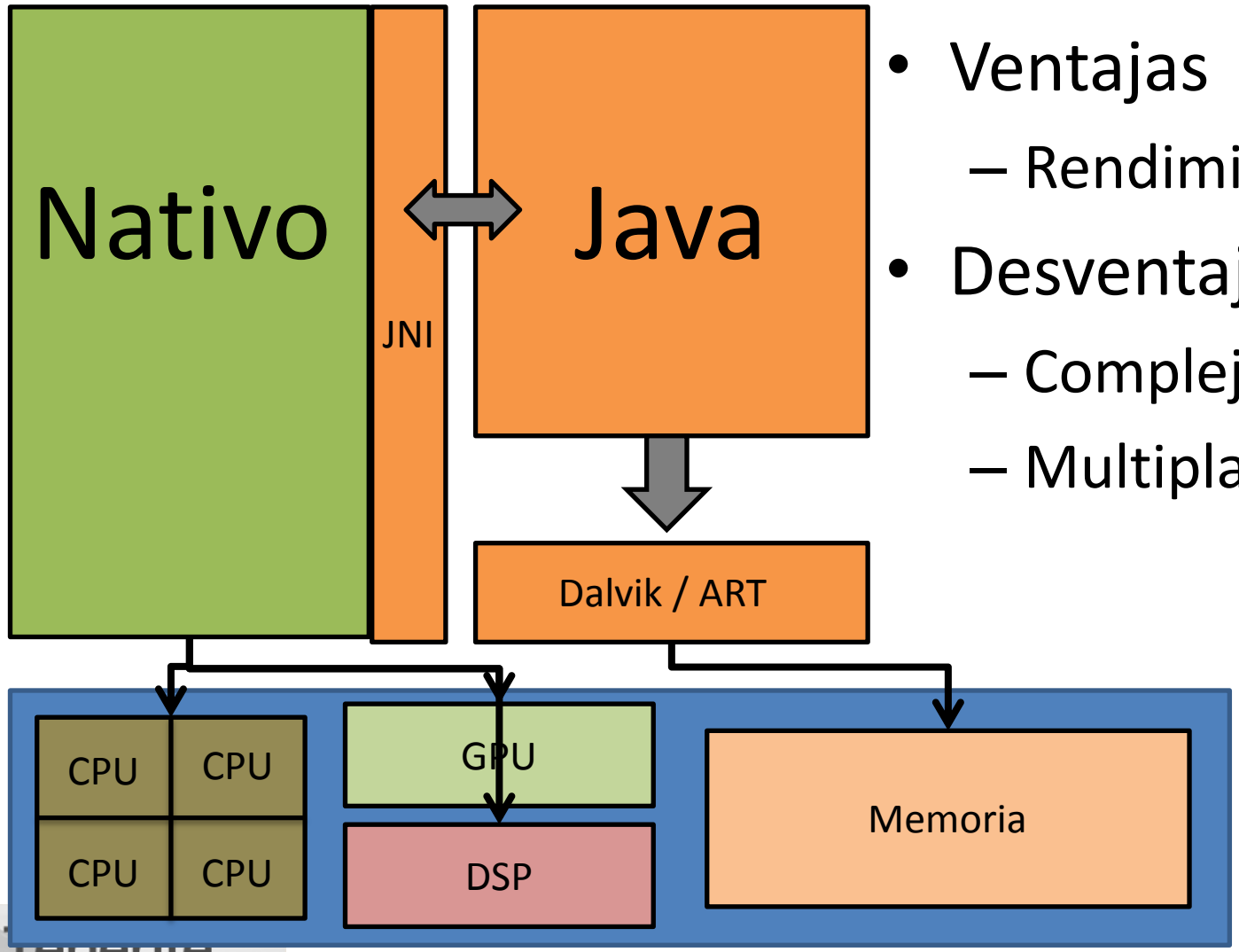
Modelos de desarrollo

Software



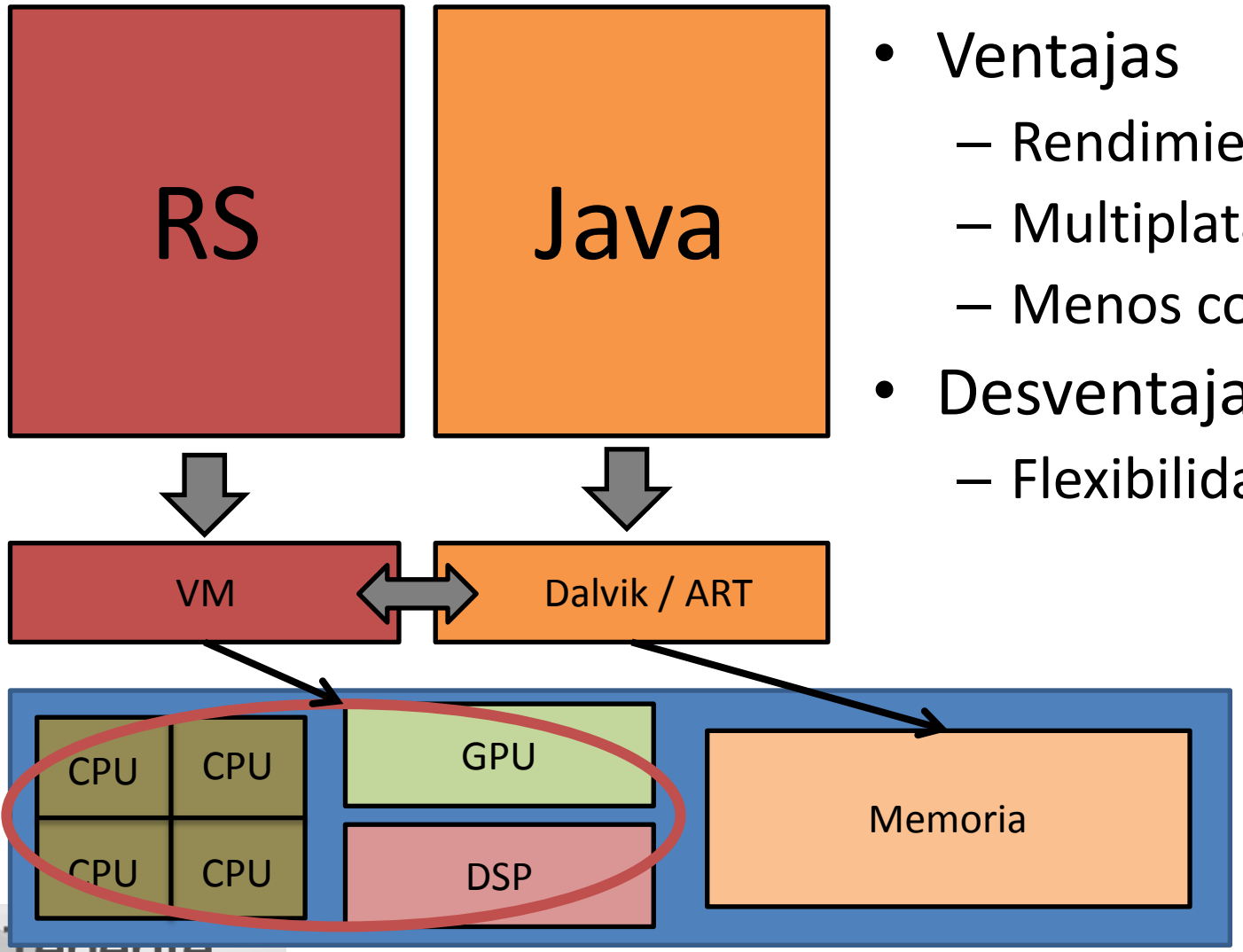
- Ventajas
  - Fácil
  - Multiplataforma
- Desventaja
  - Rendimiento

Software



- Ventajas
  - Rendimiento
- Desventaja
  - Complejo
  - Multiplataforma

Software



- Ventajas
  - Rendimiento
  - Multiplataforma
  - Menos complejo
- Desventaja
  - Flexibilidad

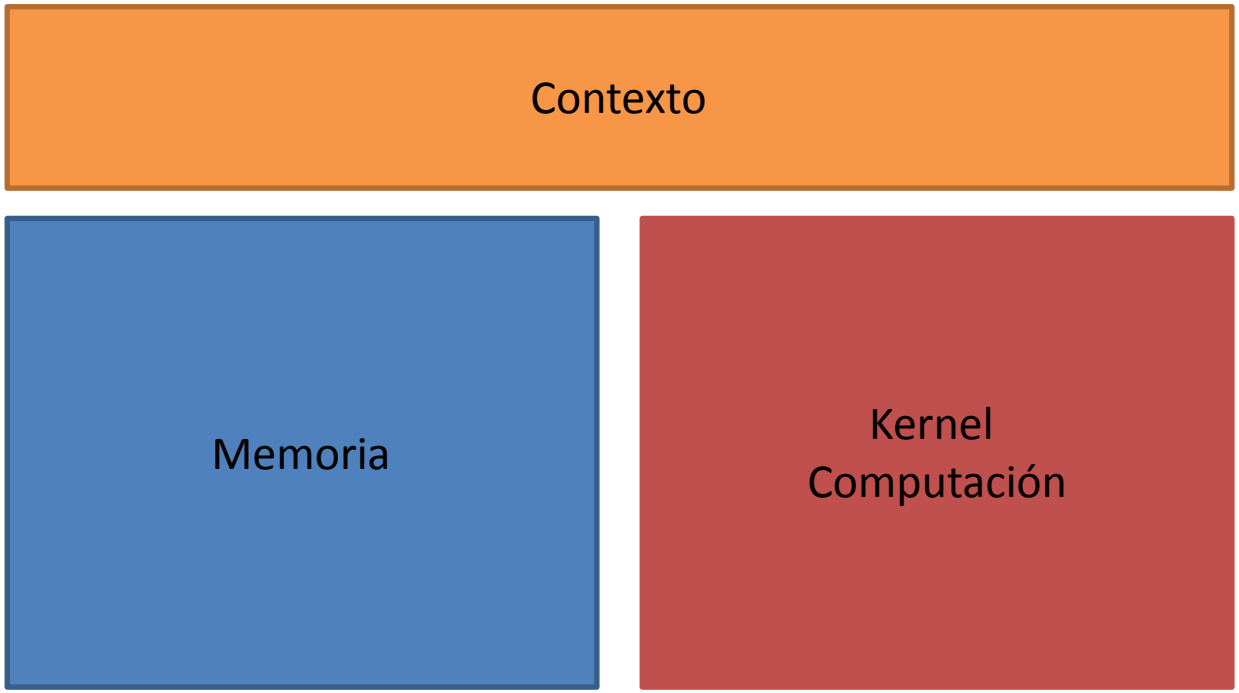
Renderscript

https://github.com/aacostad/Renderscript

Renderscript

- Android Honeycomb.
- API para computación.
- Basado en C99.
- Portabilidad como prioridad.
- Android 4.2 soporte para GPUs.
- Simplifica en lo posible el desarrollo. (Glue code).

Desarrollo

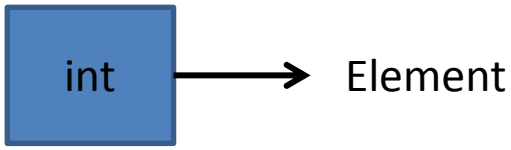


Contexto

- Comprueba que Renderscript puede ser usado.
- Puede ser una operación larga.
- Tipos de contexto
  - DEBUG
  - NORMAL
  - PROFILE
- Permite controlar el ciclo de vida de los objetos Renderscript

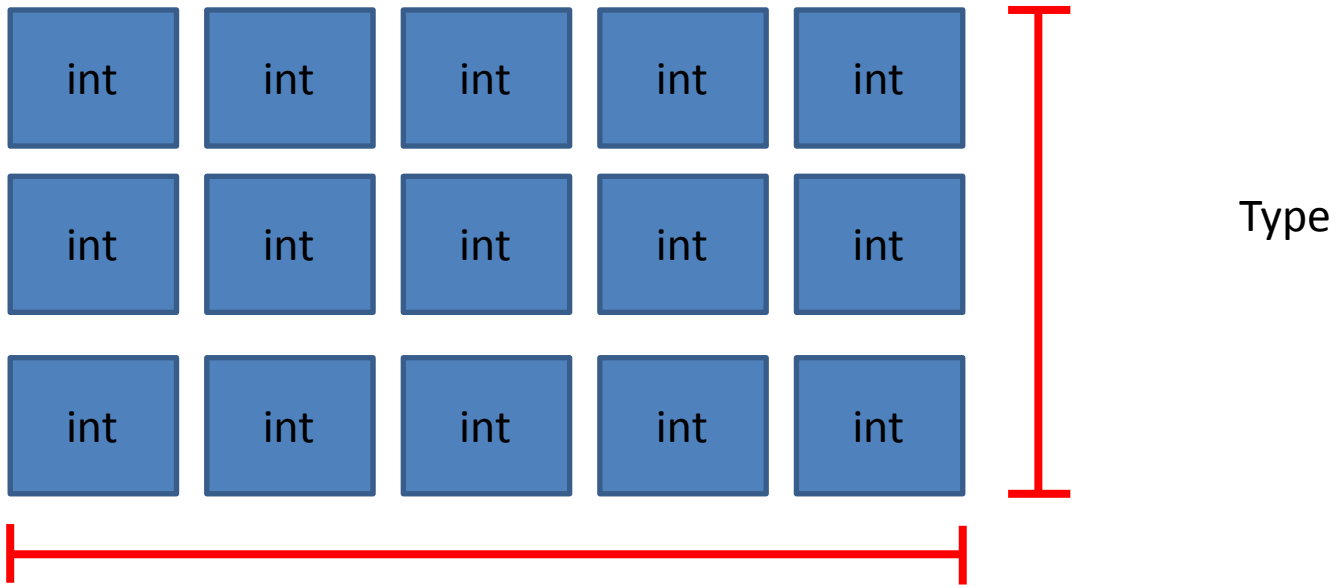
```
Renderscript mRS = Renderscript.create(Context)
```

Memoria



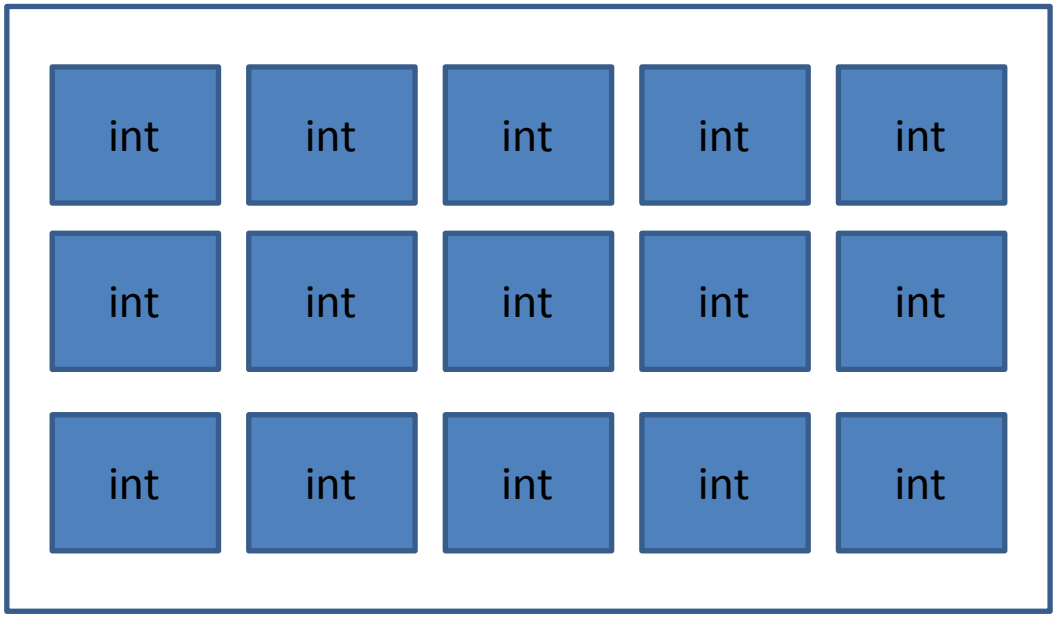
Element.I32(mRS)

Memoria




```
Type.Builder tb = new Type.Builder(mRS, Element.I32(mRS));
tb.setX(x);
tb.setY(y);
Type t = tb.create();
```

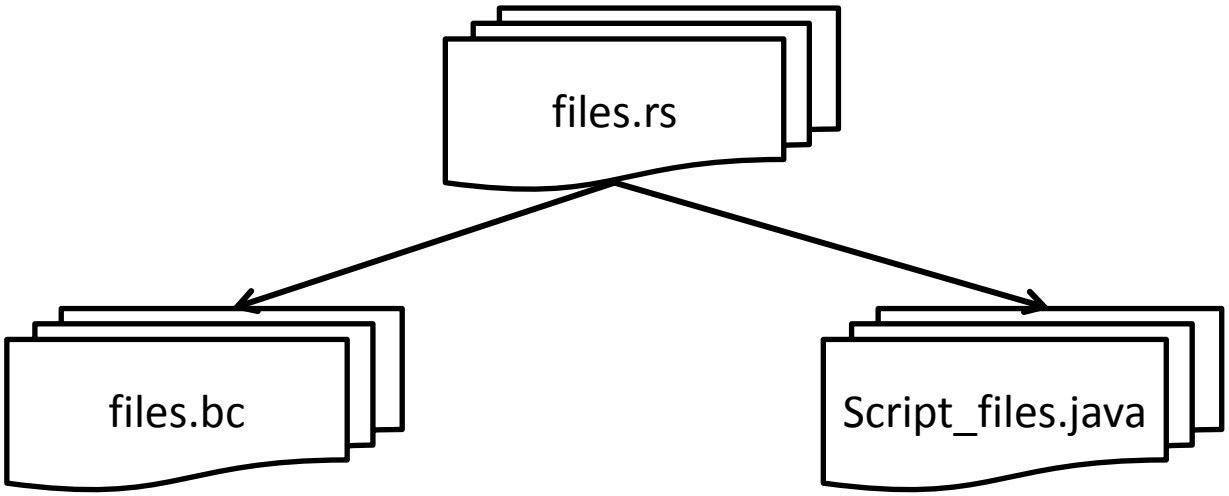
Memoria



Allocation

```
Allocation alloc = Allocation.createTyped(mRS, t);
Allocation alloc = Allocation.createSized(mRS, Element.I32(mRS), count);
Allocation alloc = Allocation.createFromBitmap(mRS, b);
```

Kernel



files.rs

```
#pragma version(1)
#pragma rs java_package_name(com.example.renderscript)

[const] [static] int var;

float4 vector;

rs_allocation in_allocation;

typedef struct Point {
    float2 position;
    float size;
} Point_t;
```

Script_files.java

- Se genera a partir del fichero .rs.
- Si las variables no son static se generan los métodos para obtener sus valores.
- Las variables static solo existen en Renderscript.
- Si se define una struct se genera una clase ScriptField_nameStruct que nos permite acceder a sus variables.

files.rs

```
#pragma version(1)
#pragma rs java_package_name(com.example.android.rs.hellocompute)


void copy (int a) {
   var = a;
}


uchar4 __attribute__((kernel)) invert(uchar4 in, uint32_t x, uint32_t y) {
     uchar4 out = in;
     out.r = 255 - in.r;
     out.g = 255 - in.g;
     out.b = 255 - in.b;
     return out;
}
```

files.rs

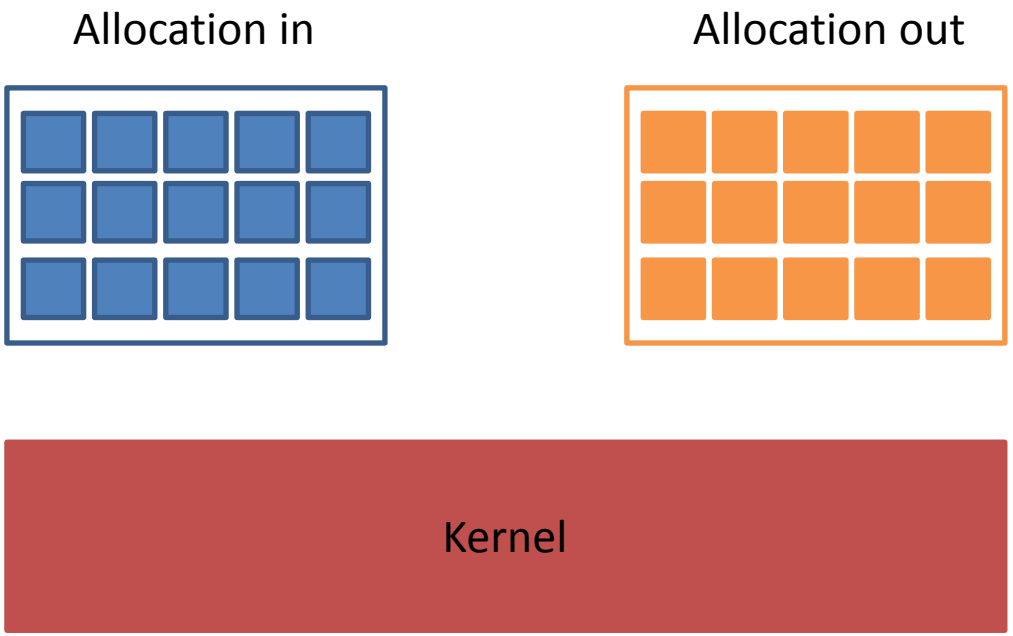
```
#pragma version(1)
#pragma rs java_package_name(com.example.android.rs.hellocompute)


void copy (int a) {
   var = a;
}
```

```
invoke_copy(int a);
```

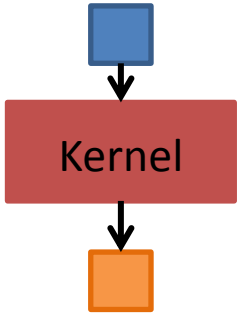
```
uchar4 __attribute__((kernel)) invert(uchar4 in, uint32_t x, uint32_t y) {
     uchar4 out = in;
     out.r = 255 - in.r;
     out.g = 255 - in.g;
     out.b = 255 - in.b;
     return out;
}
```

```
forEach_invert(Allocation in, Allocation out);
```

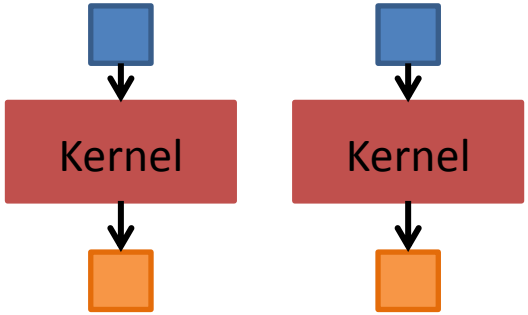
Ejecución kernel



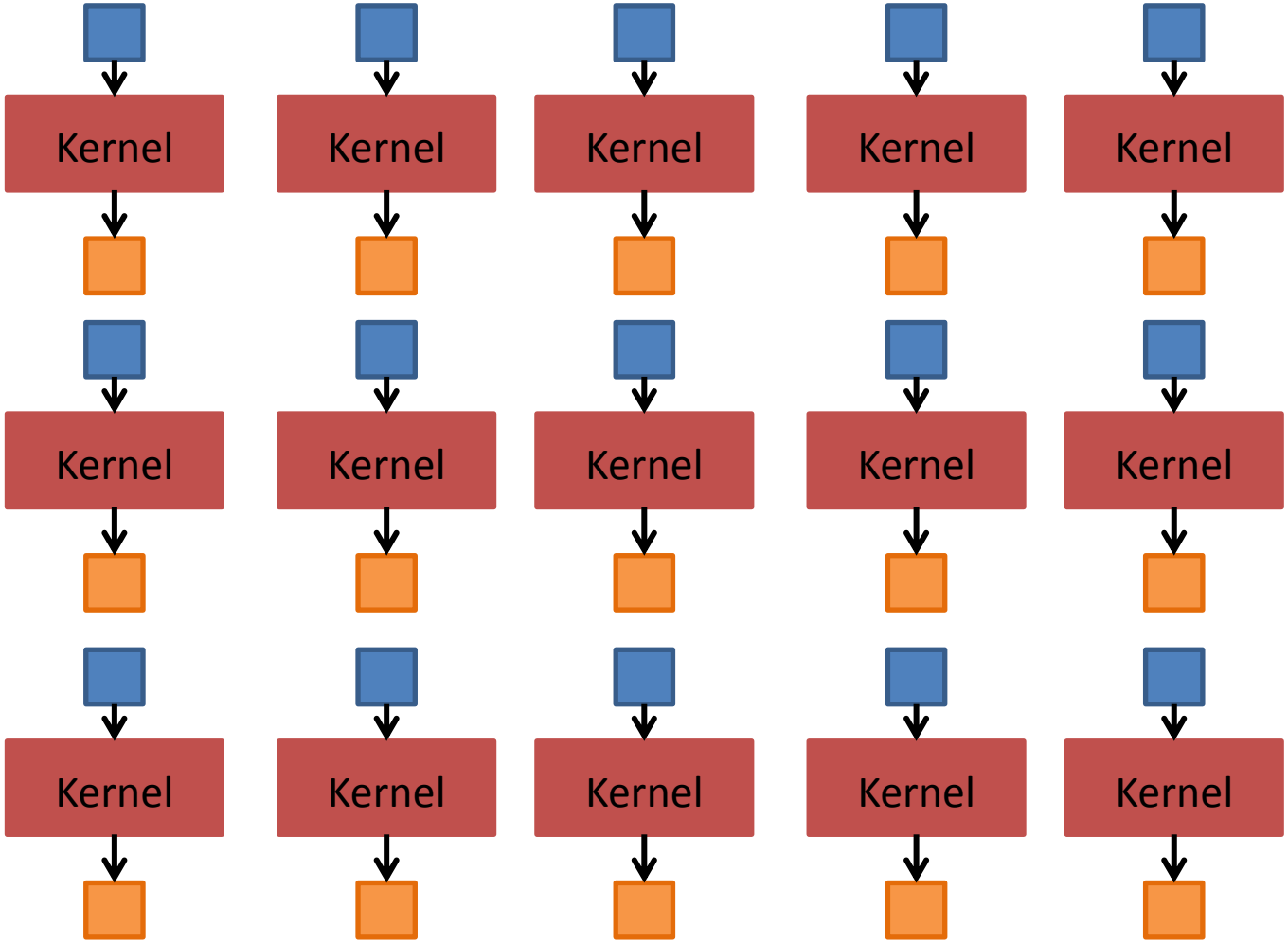
Ejecución kernel



Ejecución kernel



Ejecución kernel



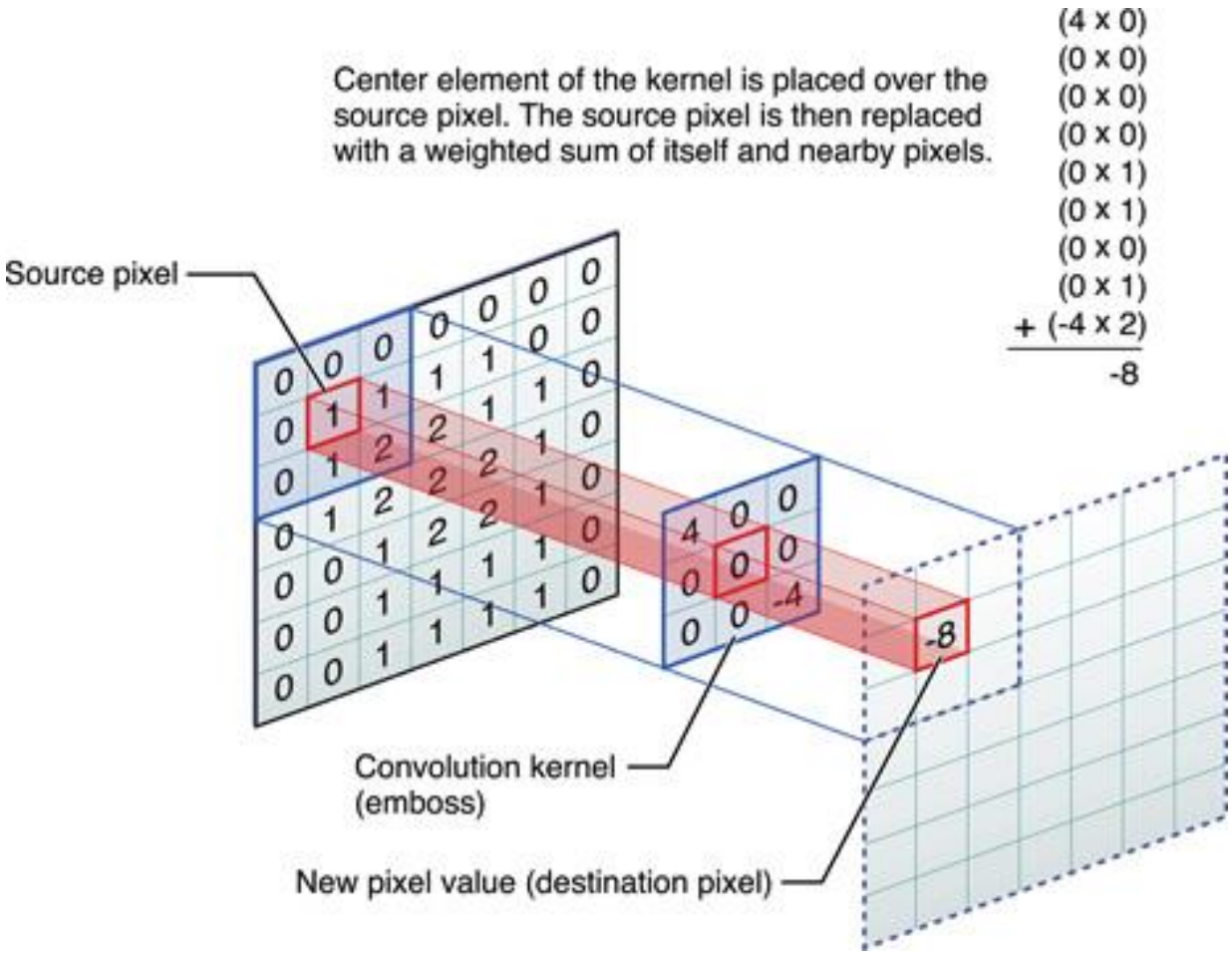
files.rs

```
#pragma version(1)
#pragma rs java_package_name(com.example.android.rs.hellocompute)


void copy (int a) {
   var = a;
}


uchar4 __attribute__((kernel)) invert(uchar4 in, uint32_t x, uint32_t y) {
     uchar4 out = in;
     out.r = 255 - in.r;
     out.g = 255 - in.g;
     out.b = 255 - in.b;
     return out;
}
```

```
invoke_copy(int a);
```

```
forEach_invert(Allocation in, Allocation out);
```

Ejemplo



https://developer.apple.com/library/ios/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

Renderscript Intrinsics

- Convolve 3x3
- Convolve 5x5
- Blur
- YuvToRGB
- ColorMatrix
- Blend
- LUT
- 3DLUT

Referencias

- http://developer.android.com/guide/topics/renderscript/compute.html
- http://developer.android.com/guide/topics/renderscript/advanced.html
- http://android-developers.blogspot.com.es/2013/08/renderscript-intrinsics.html

Acelera tus aplicaciones con Renderscript

Alejandro Acosta