Automatic Generation of OpenCL Code for ARM Architectures

Sergio Afonso
safonsof@ull.es

Alejandro Acosta
aacostad@ull.es

Francisco Almeida
falmeida@ull.es

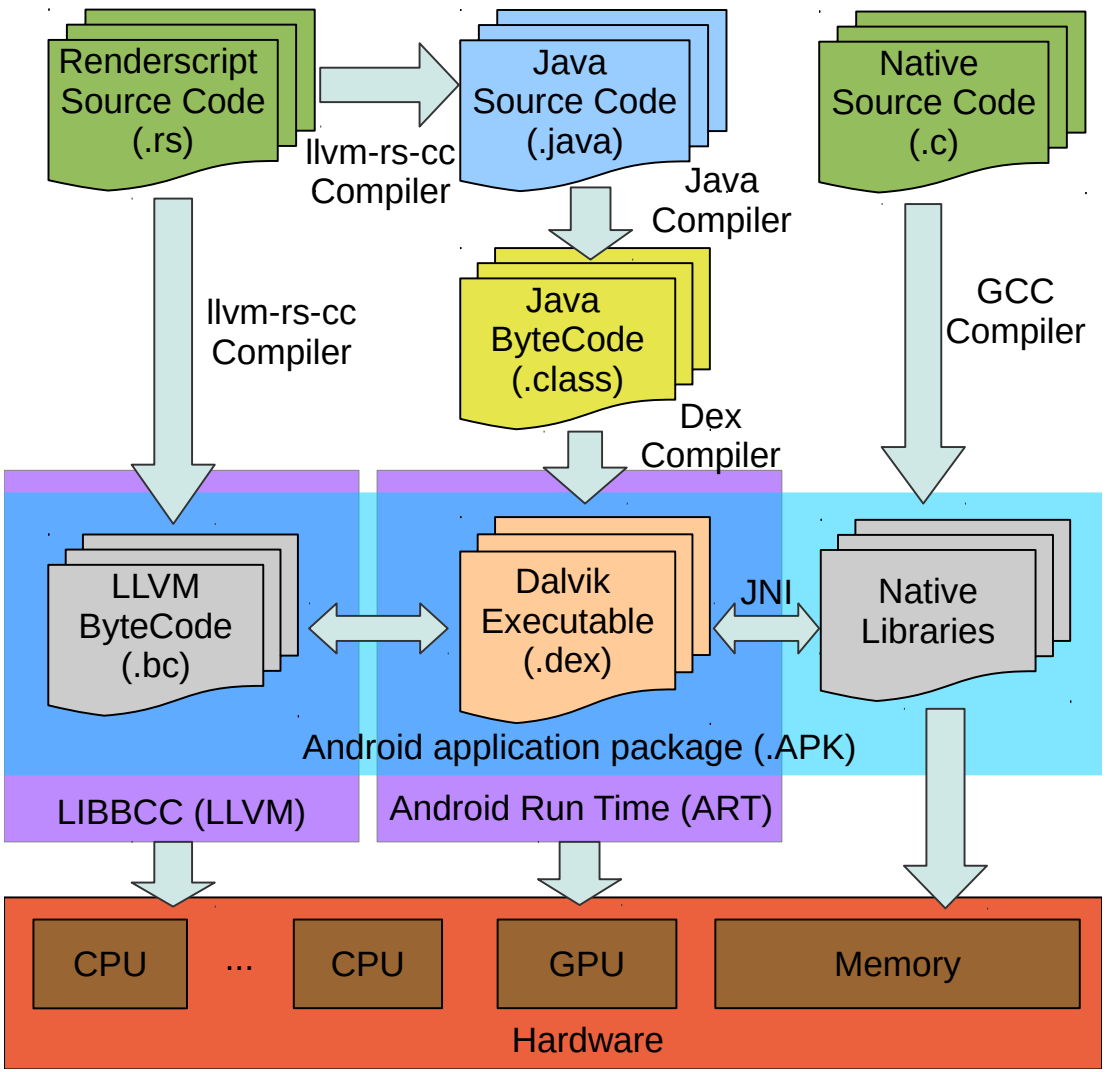
Universidad de La Laguna

High Performance Computing Group: http://cap.pcg.ull.es/

Introduction

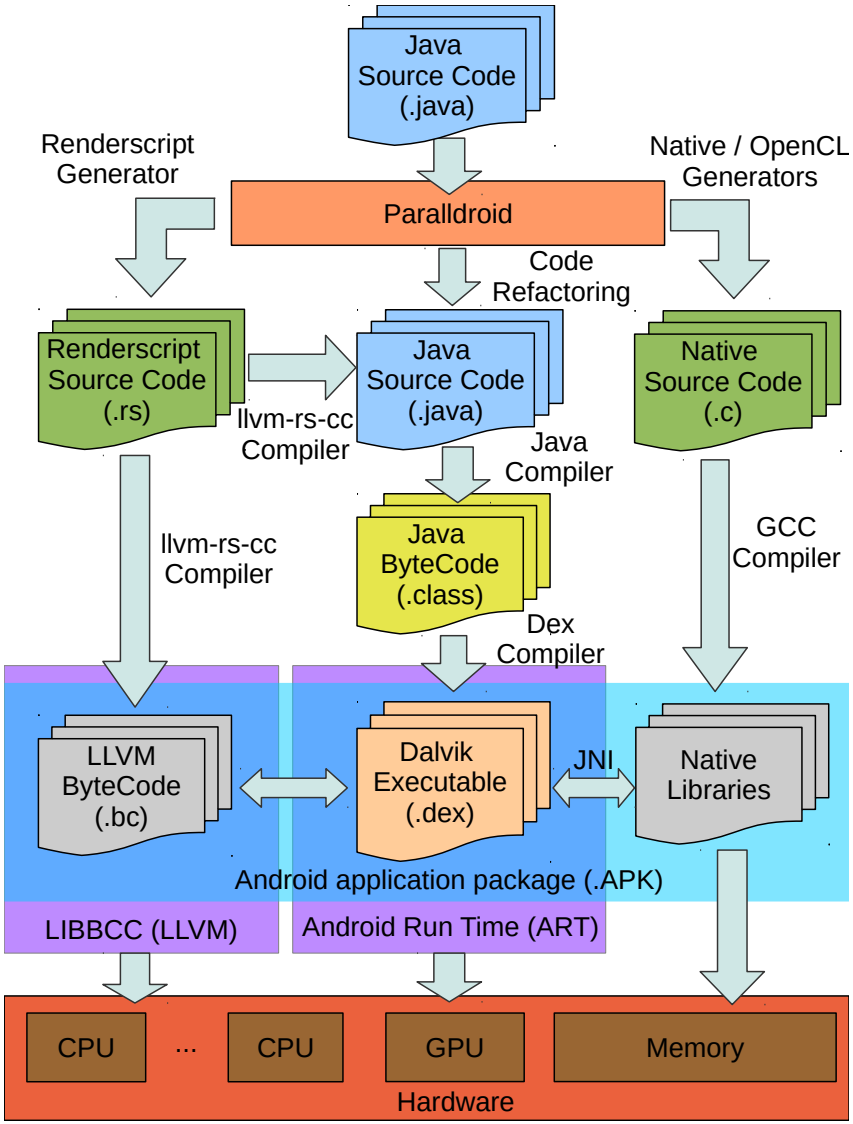
- Systems on Chip have experienced an increase of performance due to the growth of the smartphone market
- We can find heterogeneity between different devices and between processors contained in each one of them
- It is still difficult to write portable high performance code for heterogeneous platforms such as these ones
- There are tools to automatically obtain accelerated code (OpenMP, OpenACC), but they are not designed for their use in the development of mobile applications
- A great range of computer vision and image processing applications in mobile devices benefit from parallel processing

Android



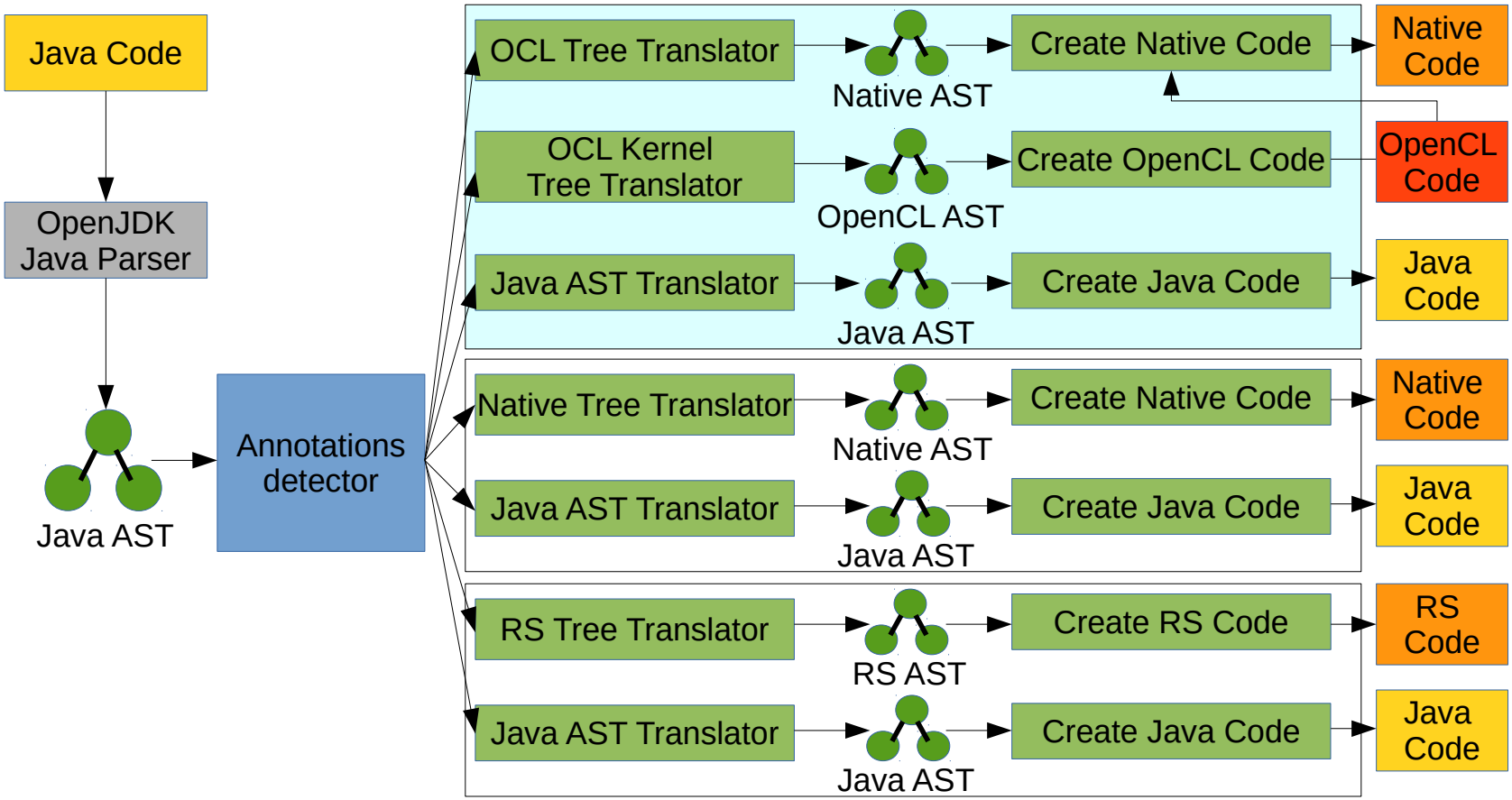
- **Java:** Main programming language of Android and easiest to program, for which an extensive range of tools is provided
- **Renderscript:** A language for high performance computing in Android, which allows data parallelism
- **Native (JNI):** Useful for reusing C/C++ code and for accessing vendor-specific native libraries

Paralldroid



- It unifies all Android programming models: It can generate Renderscript, OpenCL and native code
- It defines a set of annotations that suit more naturally a Java program than i.e. OpenMP
- It makes the development of parallel methods substantially easier than plain OpenCL and all the code needed in order to run it from Java
- It is implemented as an extension to the OpenJDK compiler

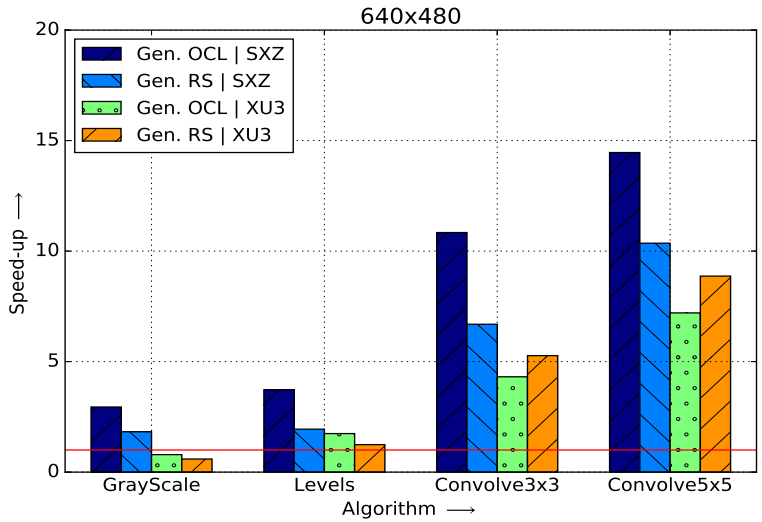
Paralldroid

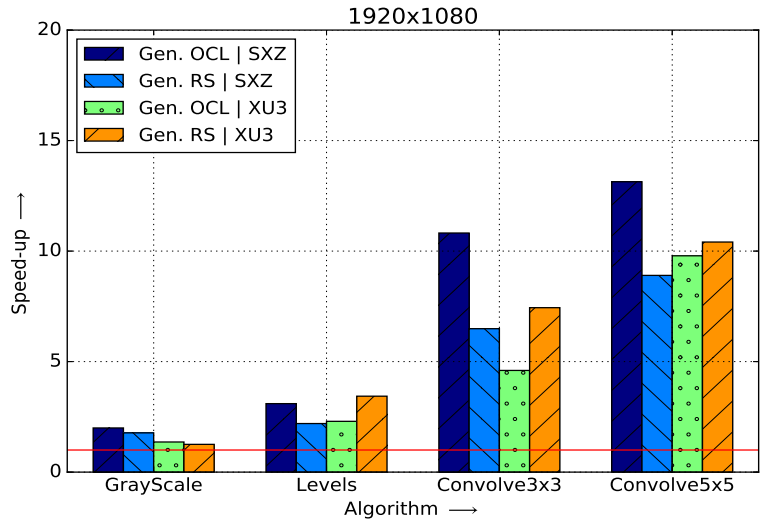


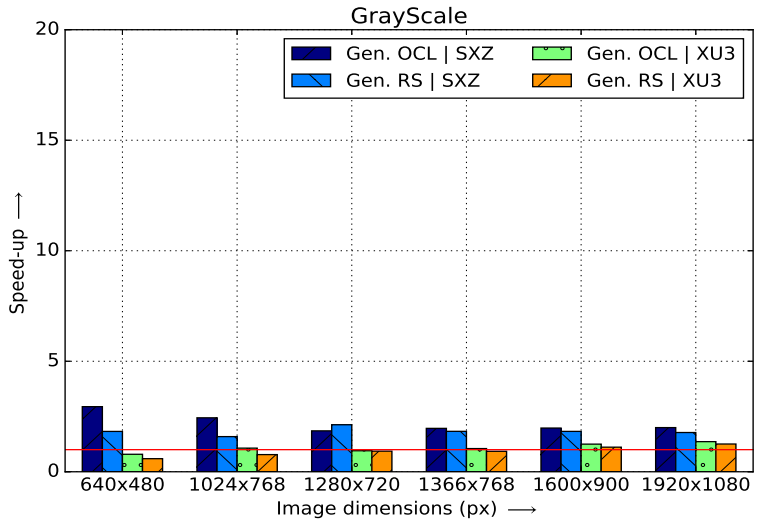
Example

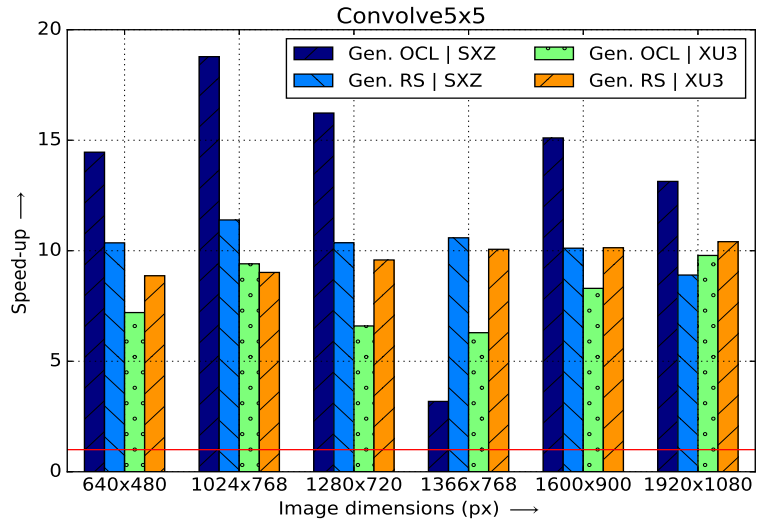
```
@Target(OPENCL)
public class GrayScale {
  @Declare
  private float gMonoMult[] = {0.299f, 0.587f, 0.114f};

  @Map(TO)
  private int width;
  @Map(TO)
  private int height;

  public GrayScale(int width, int height) {
    this.width = width;
    this.height = height;
  }

  @Parallel
  public void run(@Input Bitmap src, @Output Bitmap out, @Index int x, @Index int y) {
    int pixel = src.getPixel(x, y);
    int acc;

    acc = (int)(Color.red(pixel) * gMonoMult[0]);
    acc += (int)(Color.green(pixel) * gMonoMult[1]);
    acc += (int)(Color.blue(pixel) * gMonoMult[2]);

    out.setPixel(x, y, Color.argb(Color.alpha(pixel), acc, acc, acc));
  }
}
```

Computational results









Conclusion

- Paralldroid eases the acceleration of Android applications by means of Java annotations and AST (Abstract Syntax Tree) transformations
- Our annotations are familiar to developers that know OpenMP, but they are also adapted to the object-oriented programming paradigm
- Each Java class in an application can be implemented using a different programming language, so that each algorithm can run using the one that provides the best results transparently
- We are currently working on improving the performance of the OpenCL backend

Acknowledgement: This work was supported by the Spanish Ministry of Education and Science through the TIN2011-24598 and TIN2016-78919-R projects, the CAPAP-H network, the NESUS IC1315 COST Action and the EC (ERDF)