Llevá tu Android App al Siguiente Nivel

Albertina Durante
@albertinad16
ASDC - Argentina Software Design Center

#IntelAndroid

Agenda

Android Native Development

Técnicas de Optimización para x86

Intel XDK: Apps Híbridas

Cloud Testing

Android Native Development

Android Applications: APK

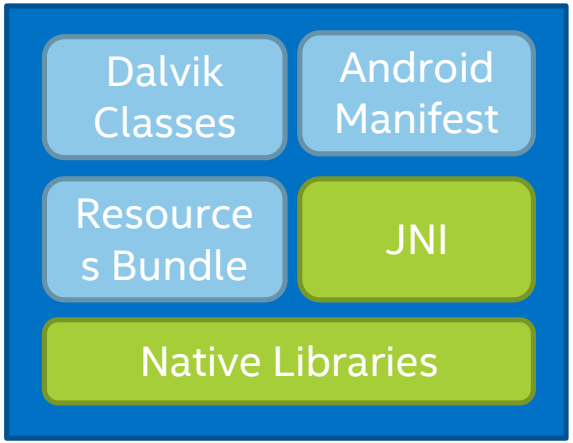
Aplicaciones Dalvik
- Código Java
- Recursos: xml, imágenes
- Android SDK

+

- Código C/C++
- Código assembly
- Librerias dinámicas .so

=

Aplicaciones NDK

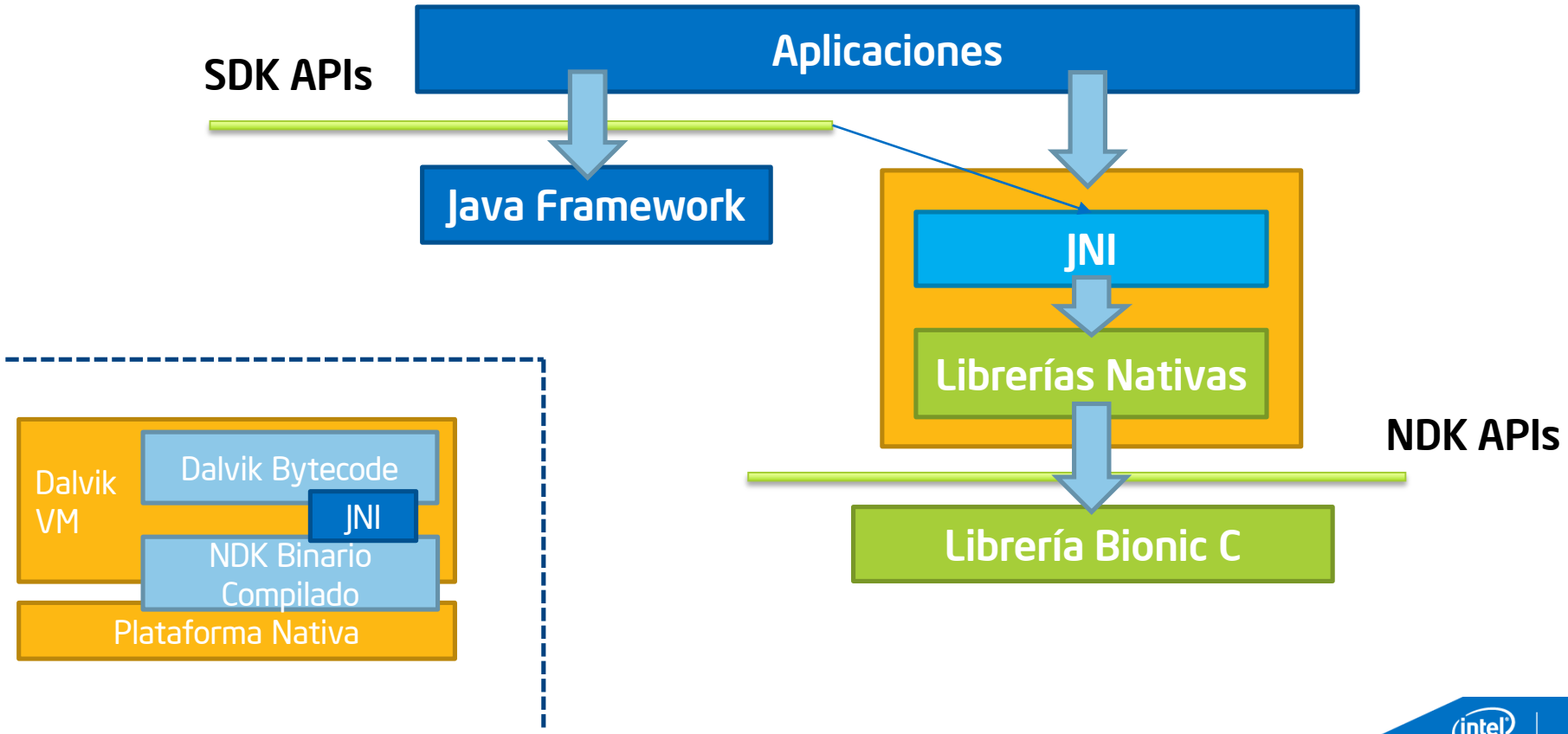


- Llamadas a funciones nativas
- Frameworks y librerías estaticas/dinámicas

Librerías nativas se compilan en binarios ".so" en libs/CPU_ABI

No existe una aplicación 100% nativa (C/C++ y assembly)

Desarrollo de Aplicaciones Android



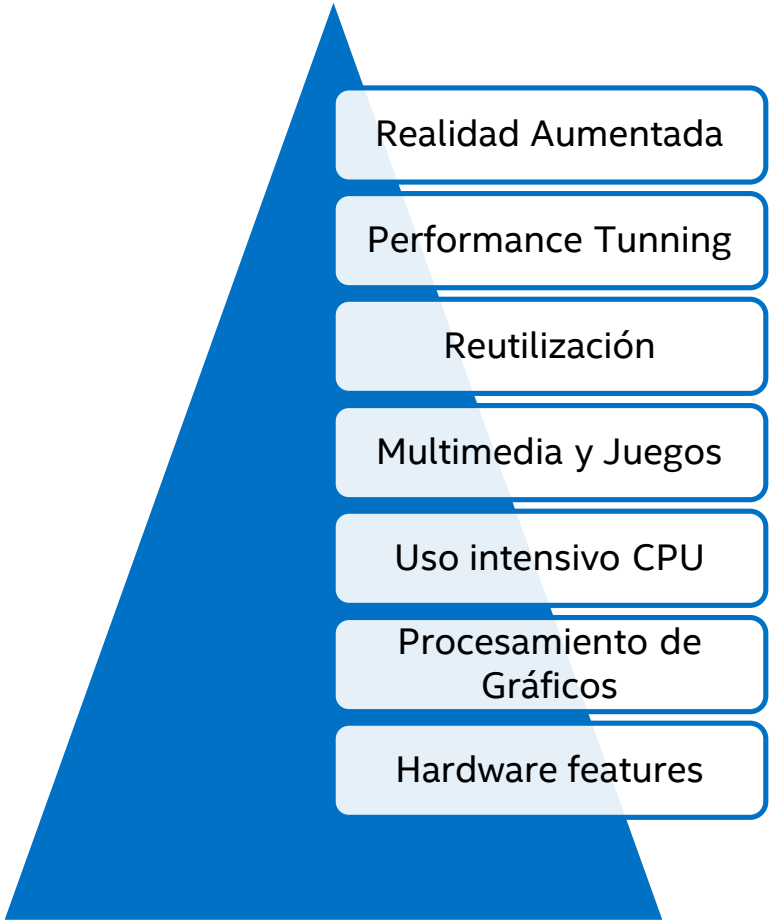
Native Development Kit (NDK)

Herramientas y build scripts que permiten implementar partes de una aplicación en código nativo como C/C++

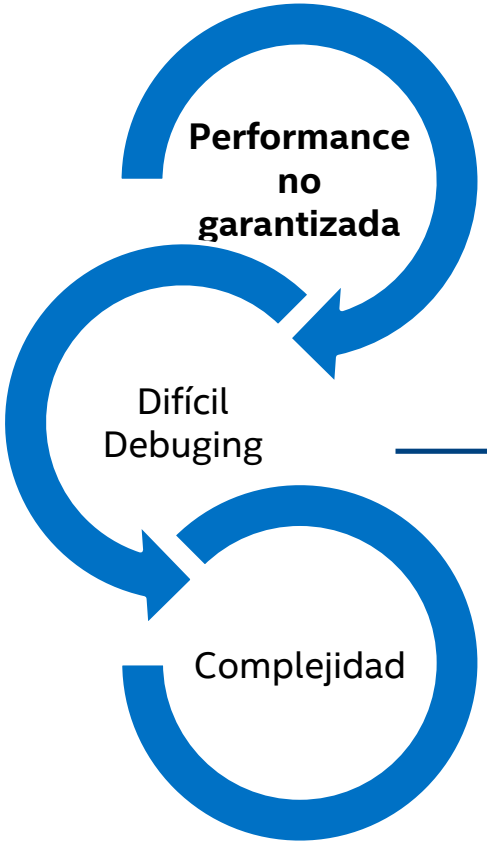
Compilar código C/C++ a librerías y ejecutables nativos específicos a la plataforma

Se debe compilar para cada plataforma que a soportar: CPU_ABI

Usos de NDK



Cuidado!



Comunicación Java y C/C++

Múltiples plataformas

Binarios para Múltiples Arquitecturas

1. Librería nativa
propia o de terceros

3. Ejecutar **ndk-build**

ndk-build APP_ABI:= x86

NDK genera código para todos los targets ABIs

2. Configuración del makefile jni/Application.mk

APP_ABI:= all
APP_ABI:= armeabi armeabi-v7a x86

Make, GCC, Intel C/C++ Compiler, Flags de optimización

Llamadas a través de JNI
C/C++ → Java
Java → C/C++

Java Native Interface

Framework que permite a código Java corriendo en una instancia de JVM ser llamado y llamar a aplicaciones y librerías nativas en lenguajes como C/C++

Estructuras de datos claves

- JavaVM: funciones de invocación, Android permite un JavaVM por proceso
- JNIEnv: JNI functions, thread-local, no se comparte entre threads.

Gestión de Memoria

- JVM: gestión de memoria de objetos Java
- Nos encargamos de las referencias a los objetos:

  slot máx de 16 referencias locales

Mapeo entre Tipos de Datos y Tipos de Signatures

Java	Nativo	Descripción
boolean	jboolean	Unsigned 8 bits
byte	jbyte	Signed 8 bits
char	jchar	Unsigned 16 bits
short	jshort	Signed 16 bits
int	jint	Signed 32 bits
long	jlong	Signed 64 bits
float	jfloat	32 bits
double	jdouble	64 bits
void	void	N/A

JNI utiliza la misma representación de Signatures que Java VM

Signature	Tipo Java
Z	boolean
B	byte
C	char
S	short
I	int
J	long
F	float
D	double

L fully-qualified-class;	fully-qualfied-class
[type	type[]
(arg-types) return-type	method type

Ejemplo:

Método Java
Double myFunction (int v1, String v2, int[] v3)

Signature
(ILjava/lang/String;[I)D

Funciones Nativas en Java

```
package com.intel.androiddev.lib;

public class NativeMessage {
     public native String getMessage();
}
```

Java keyword indicando que el siguiente **método se declara en una clase nativa C/C++**

```
static {
     System.loadLibrary("nativemessages");
}
```

Cargar la librería antes de utilizar

- System.loadLibrary
- System.load(<full_path>)

¿Cómo asociar código Java y código Nativo? javah y JNI_OnLoad

Métodos de Asociación de Código Java y Nativo

Javah

Herramienta que ayuda a generar los headers JNI a partir de una clase Java.

javah -classpath bin/classes/ -d jni/ com.intel.android.lib.Native


```
JNIEXPORT jstring JNICALL
Java_com_intel_applatina_lib_NativeMessage
_getMessage (JNIEnv *, jobject);
```

JNI_OnLoad

Método ejecutado cuando el ClassLoader instancia la clase java

Enfoque recomendado para cargar libs nativas

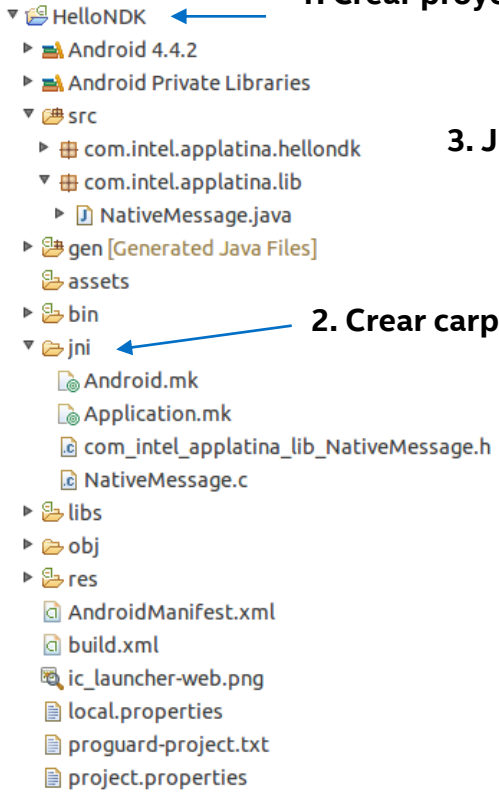
- Evita errores cuando se hace refactoring de código
- Agregar/remover funcionalidad con mayor control

```
jint JNI_OnLoad(JavaVM* vm, void* reserved) {
        ...
        JNINativeMethod methodsInfo[0];
        methodsInfo[0].name = "getCpuInfo";
        methodsInfo[0].signature = "(II)I";
        methodsInfo[0].fnPtr = getCpuInfo;

        jclass clsInfo = (*env)->FindClass(env, "com/intel/lib/NativeInfo");
        (*env)->RegisterNatives(env, clsInfo, methodsInfo, 1);

        return JNI_VERSION_1_6;
}
```

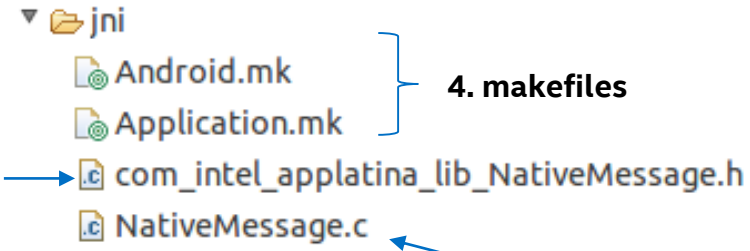
Hands-on: Hello NDK



1. Crear proyecto standard de Android

2. Crear carpeta jni para source code nativo

3. JNI header generado con javah



4. makefiles

5. Código fuente nativo

Clase NativeMessage.java

```
package com.intel.androiddev.lib;

public class NativeMessage {

    public native String getMessage();

}
```

Hands-on: Hello NDK

```
static {
    System.loadLibrary("nativemessages");
}
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    this.messages = new NativeMessage();
    this.txtvMessage = (TextView) findViewById(R.id.txtv_message);
    this.txtvMessage.setText(this.messages.getMessage());
}
```

Clase MainActivity.java

Cargar la librería nativa

Instancia de la clase NativeMessage, la cual define funciones nativas

Llamada a la función nativa

Hands-on: Hello NDK

```
#include <jni.h>

#ifndef _Included_com_intel_androiddev_lib_NativeMessage
#define _Included_com_intel_androiddev_lib_NativeMessage
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     com_intel_androiddev_lib_NativeMessage
 * Method:    getMessage
 * Signature: ()Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL Java_com_intel_androiddev_lib_NativeMessage_getMessage
  (JNIEnv *, jobject);

#ifdef __cplusplus
}
#endif
#endif
```

Header generado con javah

javah -classpath bin/classes/ -d jni/ com.intel.androiddev.lib.NativeMessage

Hands-on: Hello NDK

Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := nativemessages
LOCAL_SRC_FILES := NativeMessage.c

include $(BUILD_SHARED_LIBRARY)
```

Nombre del módulo resultado de la compilación

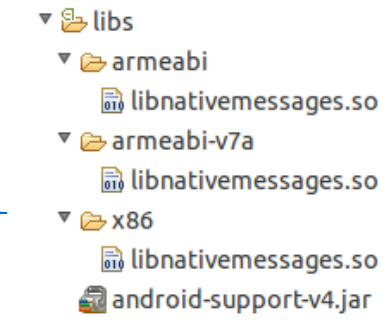
Código fuente a compilar

Application.mk

```
APP_ABI := armeabi armeabi-v7a x86
```

Tagets ABI: Application Binary Interface
Compilar para ARM y x86

```
durantea@durantea-mobl1: ~/Android-x86/workspace/HelloNDK
durantea@durantea-mobl1:~/Android-x86/workspace/HelloNDK$ ndk-build
Android NDK: WARNING: APP_PLATFORM android-19 is larger than android:minSdkVersion 10 in ./A
ndroidManifest.xml
[armeabi] Compile thumb  : nativemessages <= NativeMessage.c
[armeabi] SharedLibrary  : libnativemessages.so
[armeabi] Install        : libnativemessages.so => libs/armeabi/libnativemessages.so
[armeabi-v7a] Compile thumb  : nativemessages <= NativeMessage.c
[armeabi-v7a] SharedLibrary  : libnativemessages.so
[armeabi-v7a] Install        : libnativemessages.so => libs/armeabi-v7a/libnativemessages.so
[x86] Compile        : nativemessages <= NativeMessage.c
[x86] SharedLibrary  : libnativemessages.so
[x86] Install        : libnativemessages.so => libs/x86/libnativemessages.so
durantea@durantea-mobl1:~/Android-x86/workspace/HelloNDK$
```



Librerías nativas generadas para cada arquitectura Target con el build script **ndk-build**

Intel Binary Translator

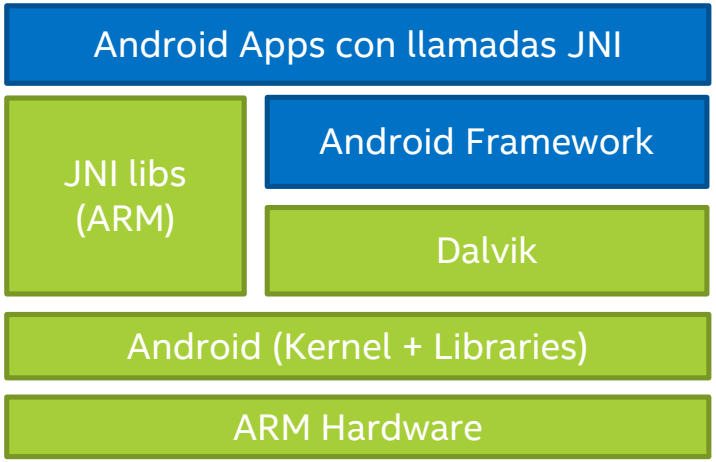
NDK Apps → Intel Atom x86

RECOMENDADO: Re-compilar con ABI x86

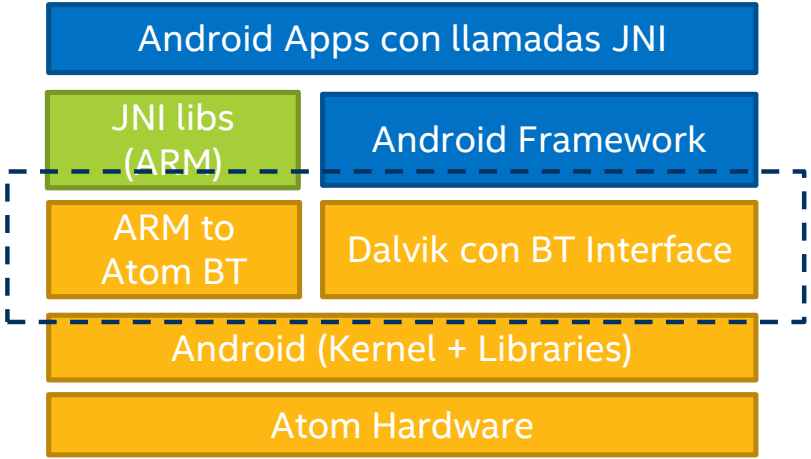
Librería que traduce código nativo ARM a código nativo x86 en runtime

Apps desarrolladas en Java se ejecutan por Dalvik, apps con libs nativas para ARM utilizan BT siendo transparente al usuario

Dispositivos ARM



Dispositivos Intel Atom



Porting y Optimización de Android Apps para Intel Atom x86

Aplicación Android

- Java y HTML5 → Dalvik App → Detectar funcionalidades que pueden ser desarrolladas con código nativo → **Performance**
- C/C++ y código **no específico** a la plataforma → NDK App
- C/C++ e instrucciones **específicas** a la plataforma → NDK ARM → Reescribir el código ARM a Intel x86
- Funciones específicas a la plataforma → NDK con librerías → ¿Librerías no compatibles con x86?
  - Open source → Recompilar la librería para Intel x86
  - Privativo → Migrar a una librería alternativa

Actualizar Target CPU

Application.mk
APP_ABI:= armeabi armeabi-v7a x86

Optimizar

GCC flags:
-march=atom
-msse4
-mavx
-maes

Compilar

ndk-build

Testear

- **Intel HAXM**
- **AppThwack**
- **Dispositivos**

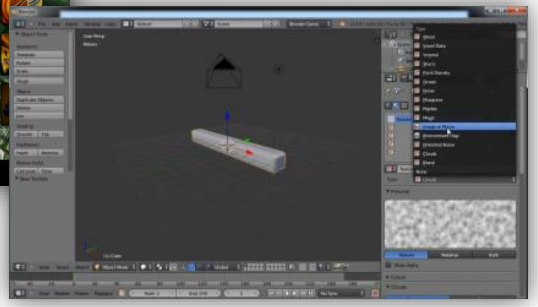
Empaquetar

APK para x86

Games Engines y Frameworks

Games Engines

- 2D Engine Cross-platform
- Open Source
- C/C++, JavaScript, Lua



Project Anarchy

- Engine para juegos mobile, para las plataformas iOS, Android, Android x86 que incluye Havok Vision Engine, Physics, Animation Studio y AI
- Arquitectura C/C++ extensible en plugins
- Optimización para rendering mobile
- Lua scripting

- 2D/3D Engine Cross platform
- Open Source
- Basado en C++ y Java
- Box2d physics

Games Engines

Soporte para x86 NEW!

Adobe AIR
release AIR 14

Ecosistema para desarrollo de juegos

- Engine de render y Herramientas
- Workflows para desarrollar contenido interactivo en 2D y 3D
- Assets disponibles

Lenguajes de Scripting

- C#, JavaScript, Boo

"[...] we are very excited to announce support for packaging of AIR applications for Intel x86 based Android devices. This support will allow AIR developers to directly target the x86 Android platform, providing the best performance possible from their AIR applications. [...]"
21/04/2014

Intel y Unity trabajando juntos para desarrollar un engine 3D optimizado para Intel x86:

- Soporte Nativo de Android para IA en todas las versiones de Unity3D
- Acceso a features únicas de gráficos Intel
- Acceso a instrucciones de IA CPU y soporte de multithreading

Apps con la mejor performance y optimizadas para Intel x86 y ARM

Frameworks

Appcelerator

Entorno de desarrollo extensible para crear aplicaciones cross-platform con código base en HTML5 + JavaScript

- **Appcelerator Platform:** enterprise platform suite: APIs, Analitics, Build, Deploy
- **Titanium:**
  - open source framework cross-platform HTML5 + JS code base
  - Módulos extensibles: incluir librería nativa optimizada para x86

Apache Cordova

- Framework para desarrollar hybrid apps cross-platform con HTML5 + JS
- Accede a features nativas de la plataforma
- Basado en plugins: extensible

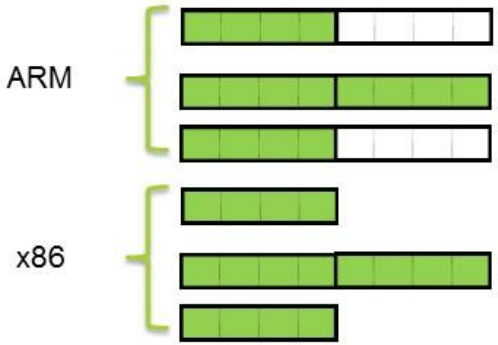
Buenas Prácticas para Desarrollo Nativo

Mejores Prácticas para Desarrollar Código Nativo

Alineación de Memoria

Por default

```
struct TestStruct {
  int var1;
  long long var2;
  int var3;
};
```



Solución de Layout de Memoria ARM ←→ Intel Atom

- Agregar “malign-double” flag al compilador GCC
- Declarar atributo con __attribute__ ((aligned(8)))

```
struct TestStruct {
  int var1;
  long long var2 __attribute__ ((aligned(8)));
  int var3;
};
```

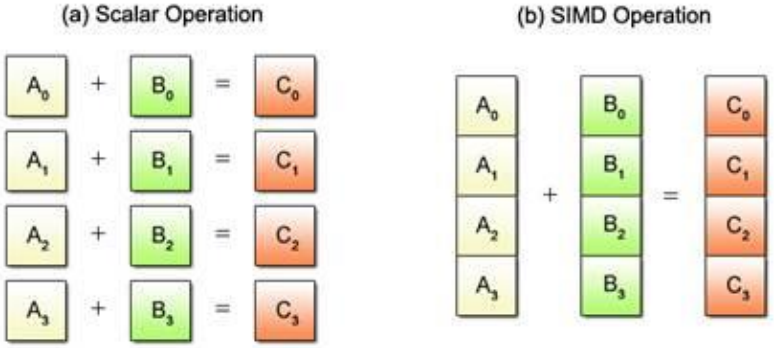
Mejores Prácticas para Desarrollar Código Nativo

Porting de instrucciones ARM NEON a instrucciones Intel SSE

Single Instrucciones Multiple Data

Técnica para lograr paralelismo a nivel de datos

Instrucciones que aplican una misma operación sobre un conjunto de grandes datos



Intel SSE

- Streaming SIMD Extension, equivalente de ARM NEON
  - SS3, SSE2, SSE3, SSSE3 (Supplemental Streaming SIMD Extension 3)
- La mayoría de las funciones de NEON tienen una equivalencia 1:1 con Intel SSE
- NEON provee librerías C nativas, deben reescribirse para ser compatibles con x86
- Intel provee un header C++ con mapping de funciones entre NEON y SSE para desarrolladores
- ***NEONtoSSE.h***

Técnicas de Optimización para x86

Técnicas Optimización

Velocidad de Ejecución → Tamaño del Código → Consumo de Energía

- Eficacia
- Igual Valor
- Combinado

Selección de instrucciones rápidas

Mejorar el grado de paralelismo

Uso efectivo de los registros de Cache

Técnicas de Optimización de la Performance

Automático por el Compilador

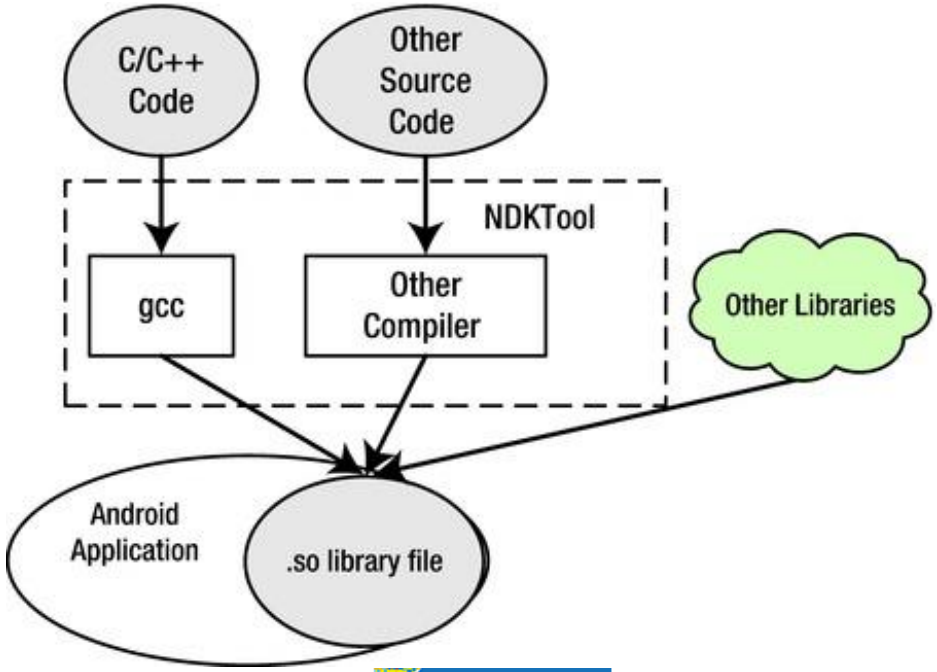
Asistencia por Tools de Desarrollo

Manual por el Developer

Compiladores

Intel C/C++ Compiler

- Utiliza features de la plataforma x86
- Código optimizado resulta un 30% más optimizado
- Basado en Intel® C/C++ Compiler XE 14.0 for Linux
- Integrado a Android NDK como toolchain adicional



Flags de optimización: independientes de la plataforma y asociadas a la plataforma

GCC Compiler: Flags de Optimización

-0 ó -01

- Reduccion de tamaño de codigo y tiempo de ejecucion

-02

- Aumenta tiempo de compilacion y performnace dle codigo generado

-03

- Activa optimizacion -02

-00

- Reduce tiempo de compilacion para debugging
- Default

Código para tipo especifico de CPU

- -march=cpu-type
- -mtune=cpu-type

Vectorización automática

- -msse, -msse2, -msse3, -mssse3, -msse4.1, -msse4.2, -msse4
- -mmmx
- -mno-sse, -mno-sse2
- -mno-mmx

Código generado para arq 32/64

- -m32-m64

-finline-functions
-funswitch-loops
-fpredictive-communing
-fgcse-after-reload

-ftree-vectorize
-fvect-cost-model
-ftree-partial-pre
-fipa-cp-clone

Recomendación

	GCC Compiler	Intel C++ Compiler
Nivel de Optimización	*-O2* o superior, *-Ofast* for peak	*-O2*, *-fast* for peak (implica-static)
Arquitectura	*-march=atom –mtune=atom –mssse3* para Atom *-march=slm –mtune=slm –msse4.2* para Silvermont *-march=atom* activa *-mmovbe (no soportado en todas las plataformas x86)* Para evitar agregar *–mno-movbe*.	*-xATOM_SSSE3* para Atom *-xATOM_SSE4.2* para Silvermont
Math	*-ffast-math* – Más rápido, menos preciso *-mfpmath=sse* – Usar SSE para cálculos FP en lugar de i387	*-no-prec-div* – Más rápido, menos preciso *-mfpmath=sse* – Usar SSE para cálculos FP en lugar de i387
Mayor Performance	*-flto* *-funroll-loops*	*-O3 -ansi-alias -ipo-auto-p32 -parallel*

Vectorización

- Loop-unrolling y generación avanzada de instrucciones SIMD
- Tarea manual del developer: no es escalable e implica costo de adaptación para cada arquitectura
- Auto-vectorización realizada por el compilador

```
APP_CFLAGS := -O3 -xSSSE3_ATOM  -vec-report3
```

Targets de Compilación

Proceso de Build de NDK:

- Se evalúa el make file Android.mk para cada arquitectura
- *TARGET_ARCH_ABI: arquitectura actual*

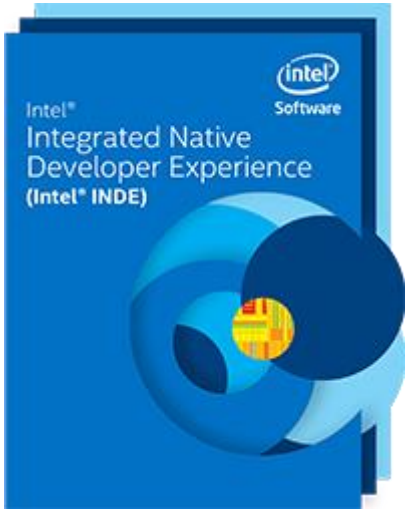
```
ifeq ($(TARGET_ARCH_ABI),x86)
LOCAL_CFLAGS := -mtune=atom -mssse3
endif

ifeq ($(TARGET_ARCH_ABI),armeabi-v7a)
LOCAL_CFLAGS := -march=armv7-a
endif
```

Application.mk

```
1 APP_ABI := armeabi armeabi-v7a x86
2
3 # Use Intel C++ Compiler for Android when compiling for x86 ABI.
4 # Intel C++ Compiler must be installed and available in the host machine
5 # (development system).
6 #
7 ifeq ($(TARGET_ARCH),x86)
8    NDK_TOOLCHAIN := x86-icc
9    APP_OPTIM=release
10   APP_CFLAGS := -xATOM_SSSE3 -O3 -vec-report3 -ipo -parallel -ansi-alias
11 endif
```

Intel Integrated Native Development Experience



Suite de desarrollo nativo cross-platform (Intel Architecture y ARM)

Cross platform meets native performance

Intel®
INDE

Cross-OS, Cross-Architecture, Cross-IDE

- Tools nativas para C/C++ y Java
- Tools integradas a IDEs popupales
- Ejemplos para Android y Microsoft Windows

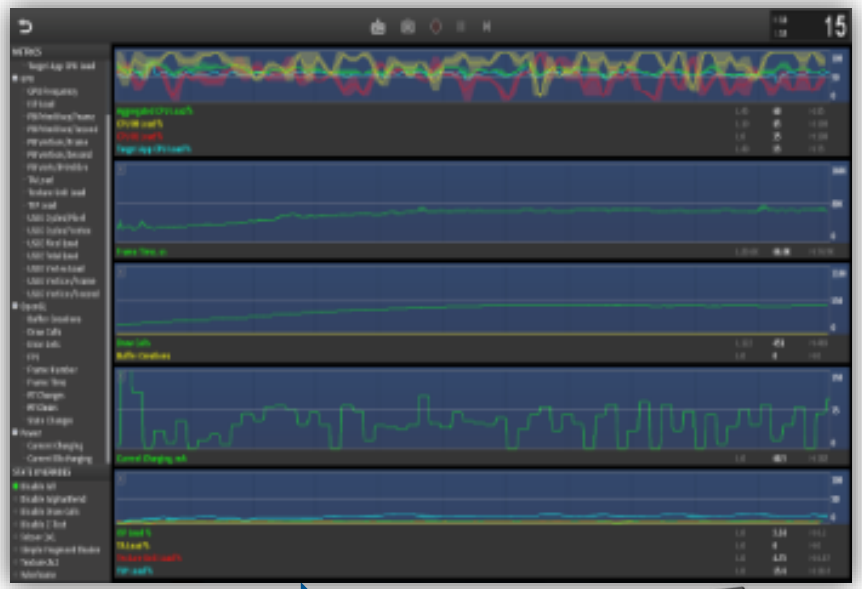
Mayor Performance, Menor tiempo

INDE Inside

- Media for Mobile
- Intel Media SDK
- Intel® Threading Building Blocks
- Intel Integrated Performance Primitives
- Intel® C/C++ Compiler y GNU C/C++ Compiler
- Compute Code Builder: soporta APIs de Google Renderscript* and OpenCL™
- Intel HAXM
- Analyzing and Debugging: Intel® GPA

Intel® Graphics Performance Analyzer Tool

- Análisis de performance en tiempo real a nivel de sistema para dispositivos basados en Android x86
- Realizar experimentos y aislar problemas de performance de CPU y GPU
- Métricas de CPU, GPU, API, memoria, red, alimentación



Android x86 Device

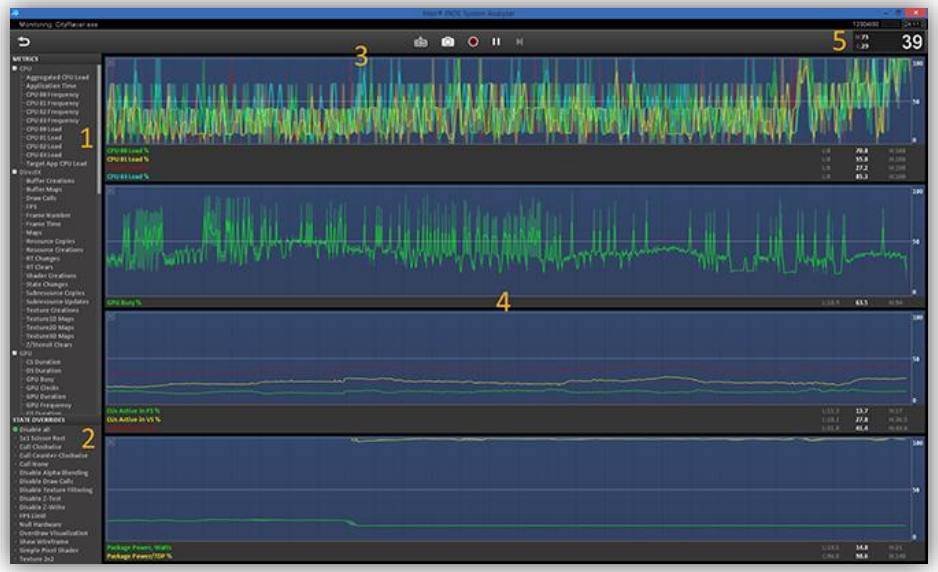
Datos

Conexión USB ADB

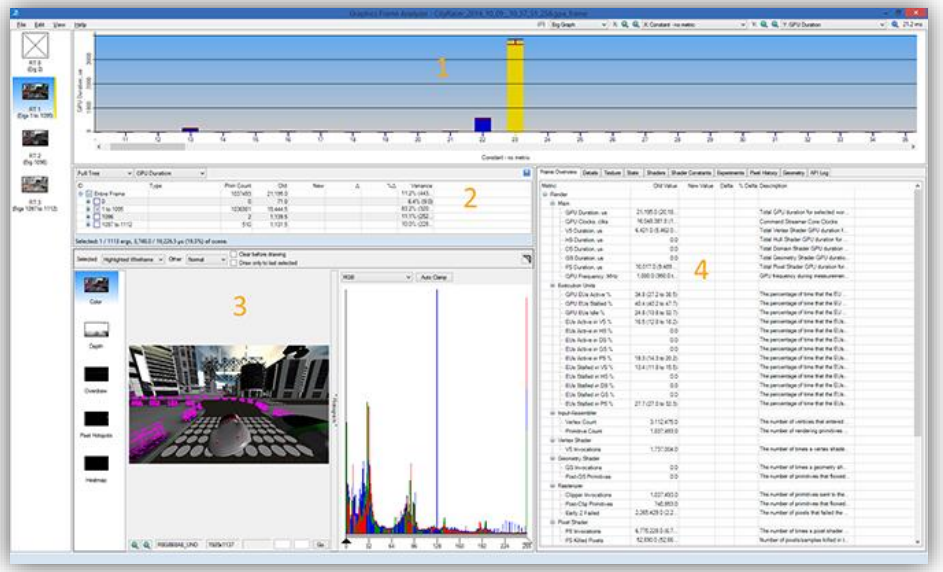
Conexión WiFi

Intel GPA System

Intel® Graphics Performance Analyzer Tool

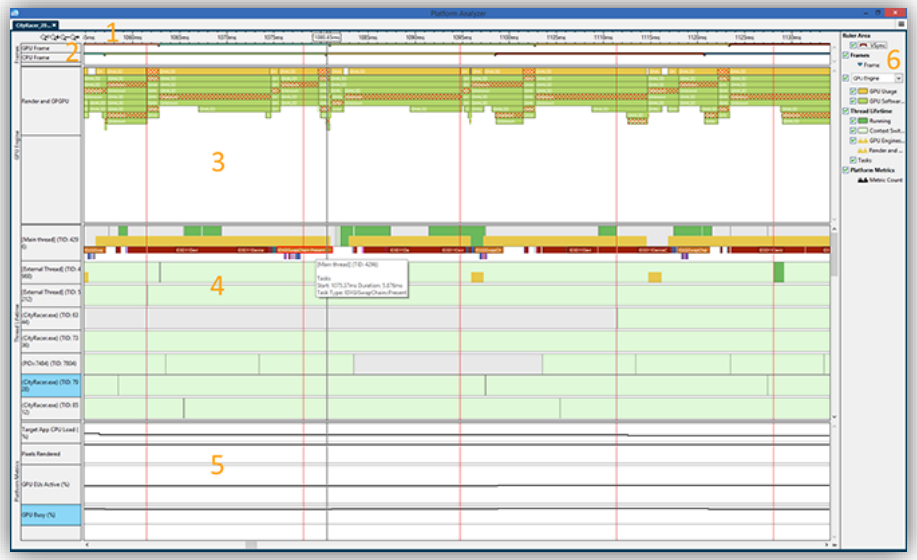


System Analyzer
Métricas de CPU, Graphics API, GPU y consumo de energía

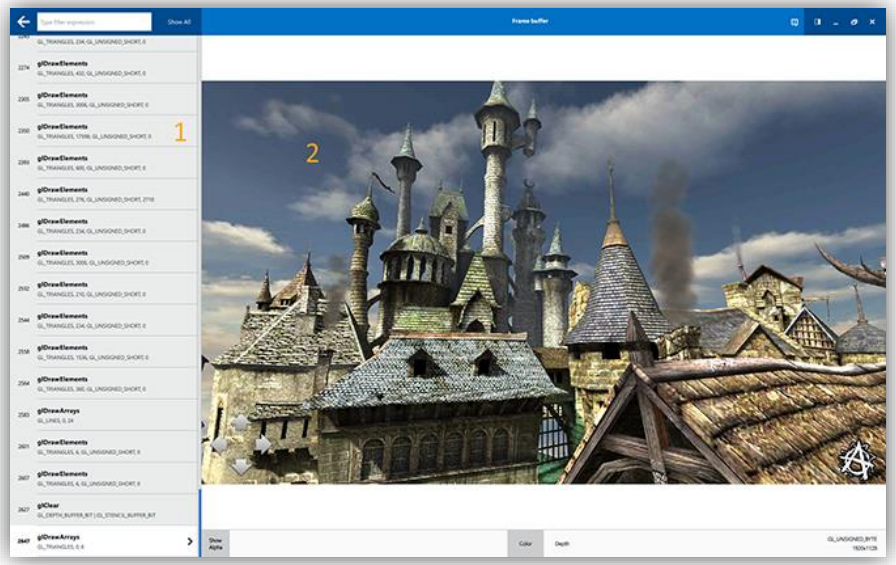


Graphics Frame Analyzer
Single-frame analysis and optimization tool for Microsoft DirectX* and OpenGL ES* game workloads

Intel® Graphics Performance Analyzer Tool



Platform Analyzer
View where your application is spending time across the CPU and GPU

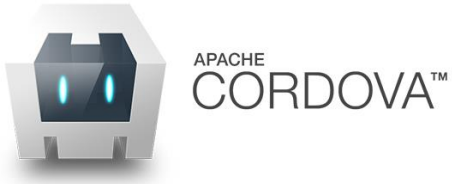


Graphics Frame Debugger
Identify rendering problems in games, track down errors, and identify complex state-related and frame content problems

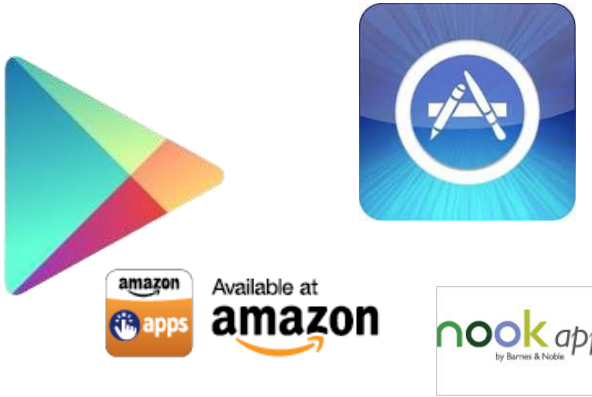
Intel XDK: Aplicaciones Híbridas

Hybrid HTML5 Apps…

…allow developers to build apps using these skills and tools…

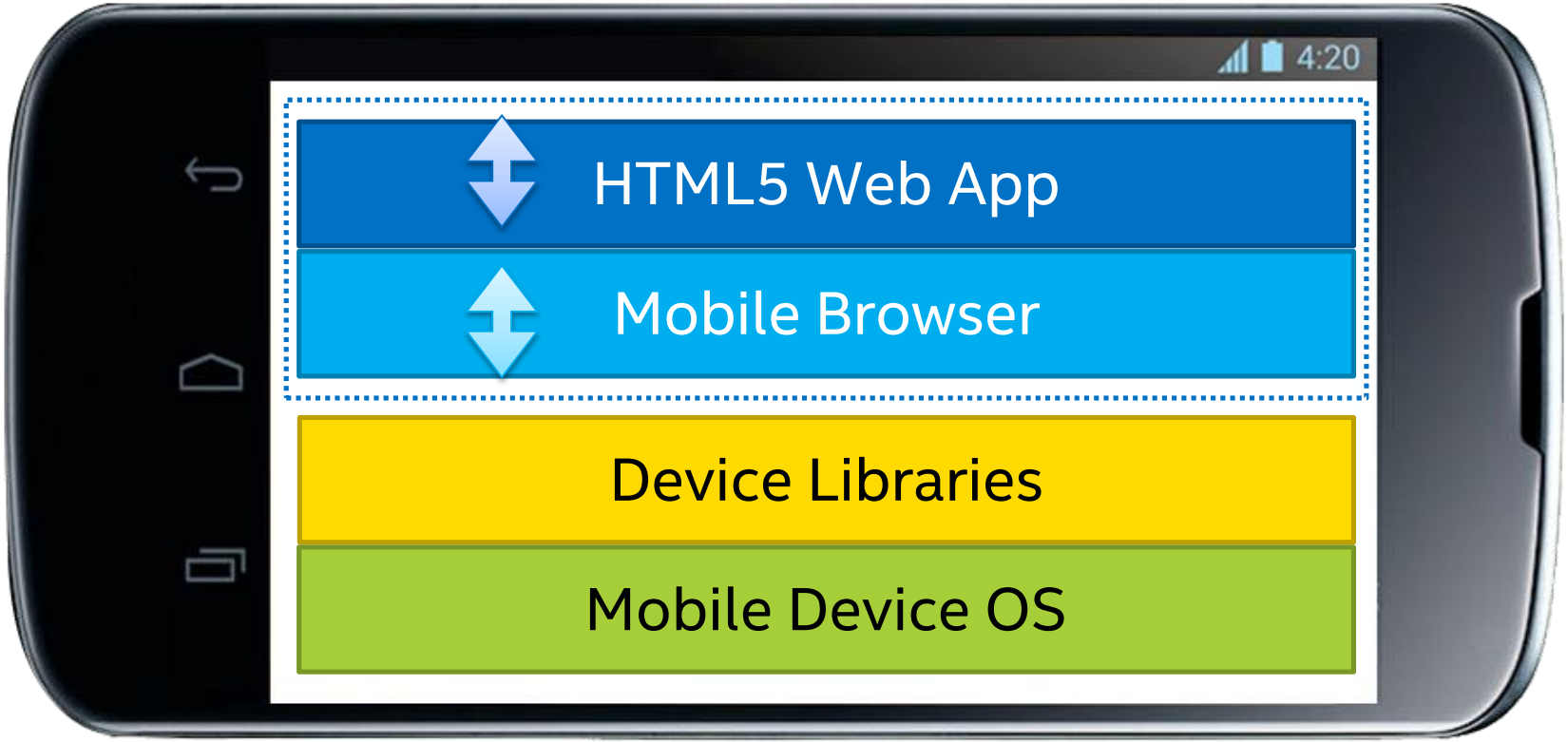


Accessing Native Device functionality

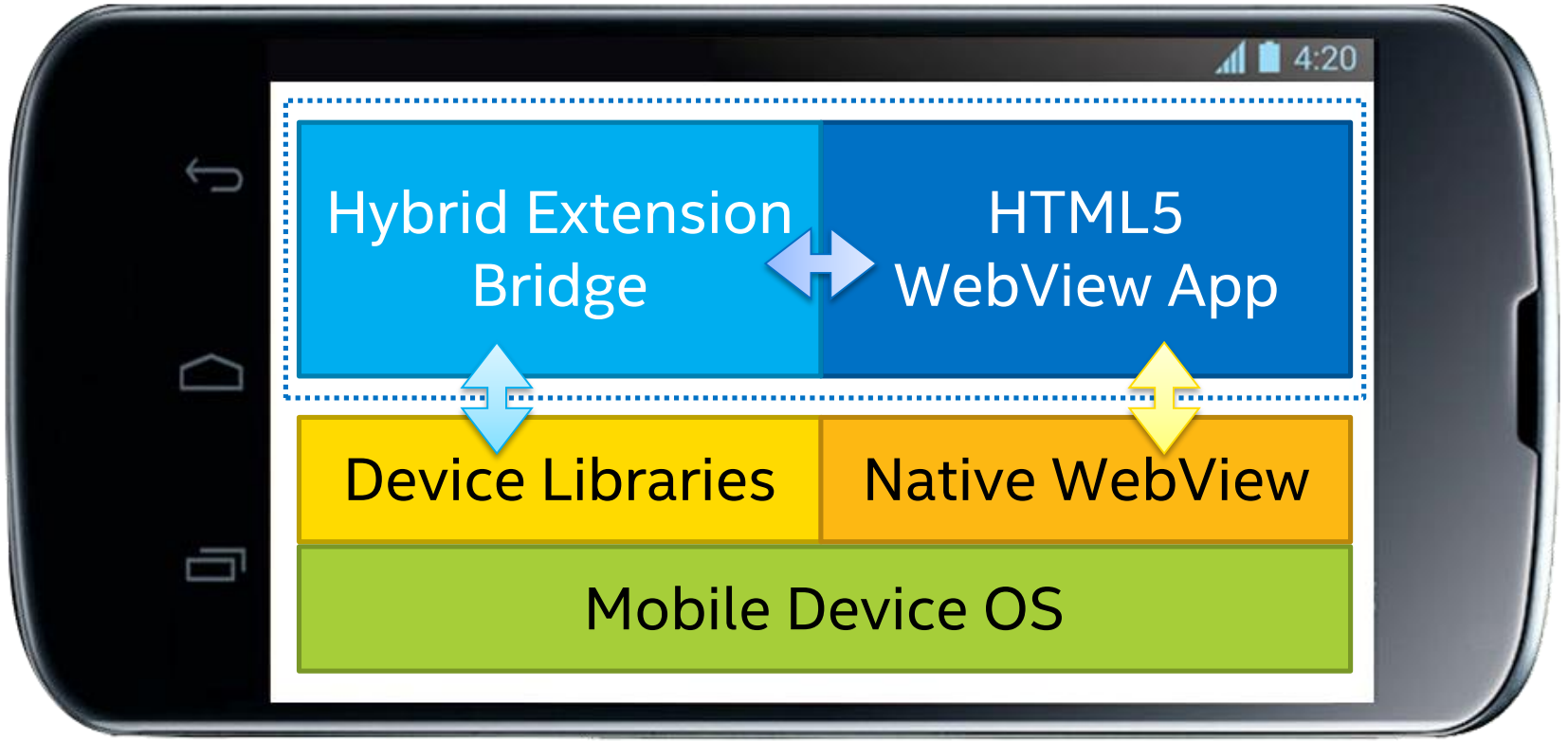


…that can be distributed in native app stores.

Mobile HTML5 Web App Block Diagram



Mobile Hybrid HTML5 WebView App Block Diagram



Web, Native & Hybrid Apps

Web

- Web Standard technologies: HTML5 + JS + CSS
- Write-once-runs-anywhere approach: cross-platform between devices
- Runs in a browser
- Limitations: devices capabilities access, secured storage, etc

Native

- Specific to a given Platform
- Language and Dev tools supported by the Platform
- Perfomance
- Device features access
- Platform UX guidelines

Hybrid

- HTML5 + JS + CSS
- Native Container
- Device feature access
- Cross-platform
- **Package and Install as an App**
- **Distribute and Publish to the Stores**

Intel® XDK

Intel® XDK free at

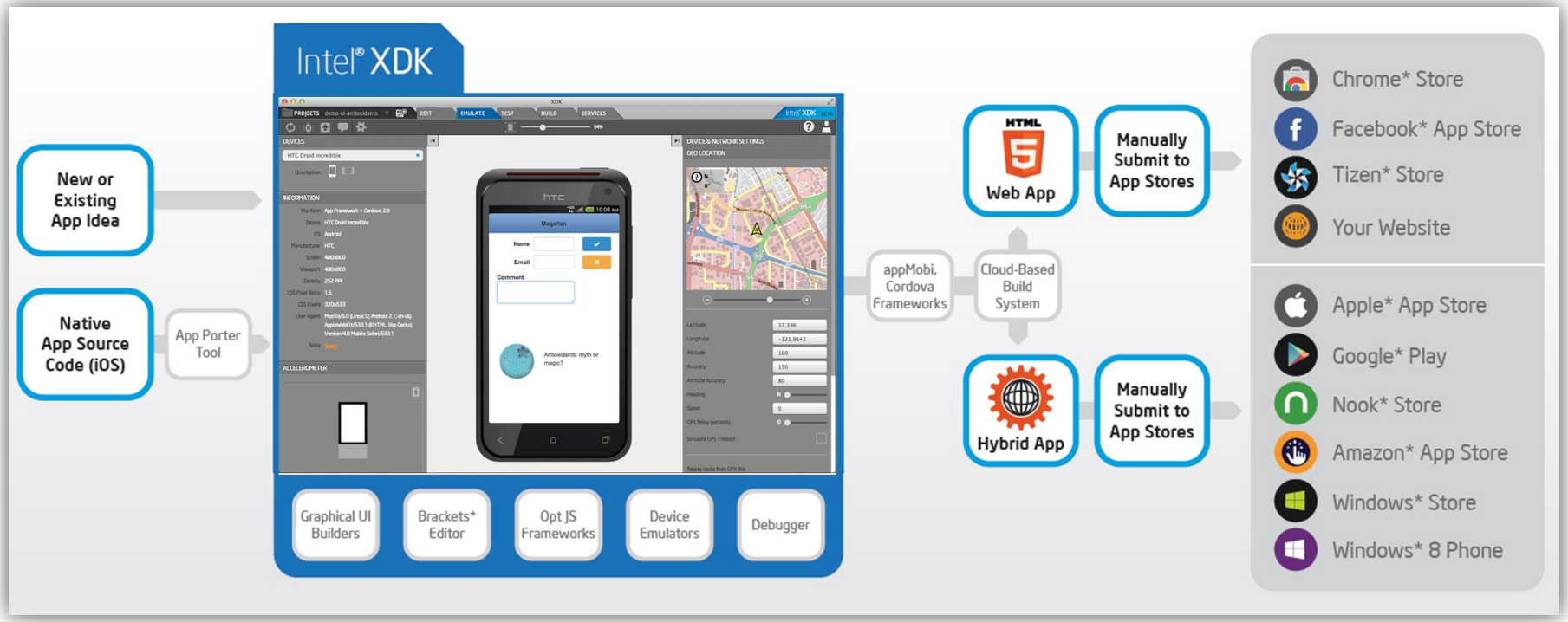
http://xdk.intel.com

Intel® XDK enables software developers to develop, test and build HTML5 web and hybrid apps across platforms, app stores and multiple form factors

- **Write Once, Run Anywhere**
  HTML5 Web Apps, Hybrid Apps
- **Faster-Time-To-Market**
  Integrated Front-To-End Tools Solution
- **Amazing App Experience**
  Optimized UI/UX JS Libraries, Performance Profiling Tools
- **Short Learning Curve**
  Simplified Workflow

intel

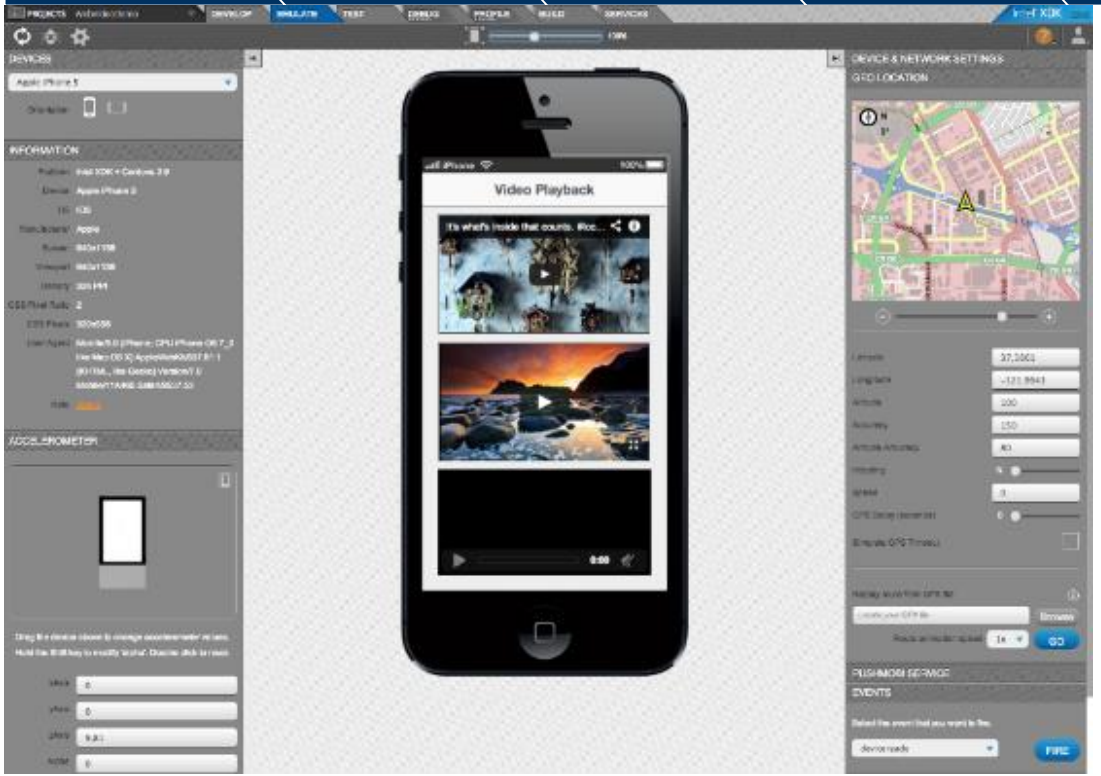
Intel® XDK and Cordova!



Intel® XDK Components

APACHE CORDOVA™

DEVELOP | EMULATE | TEST | DEBUG | PROFILE | BUILD



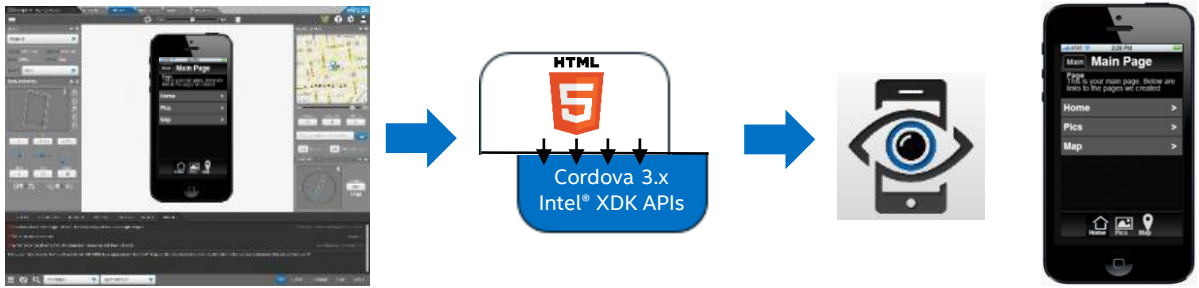
- Accelerometer
- Camera
- Compass
- Contacts
- Geolocation
- Device
- Notification
- Storage

Display, Multitouch, Connection, Event,s File, Globalization, Media InAppBrowser, and more...

App Preview

Your HTML5 App
with Cordova API calls

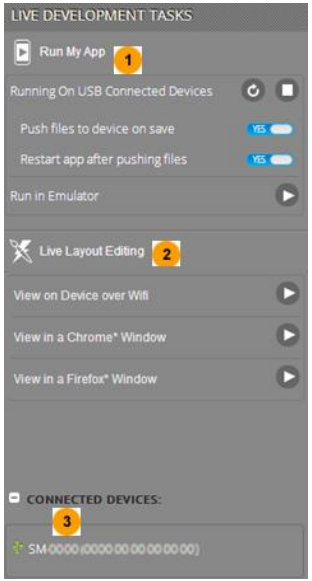
App Preview
with Cordova runtime



- Native Container/WebView
- Closes the gap between app development and on-device testing

Testing of hybrid apps on real devices without going through app store submission processes – for faster TTM

Live Development

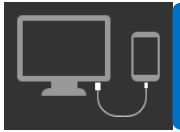


Live Layout Editing

View your app on WiFi-connected Android and/or Apple iOS* device(s), or in a browser window

Changes appear immediately after you make edits using the built-in Intel XDK editor

Remote Debugger



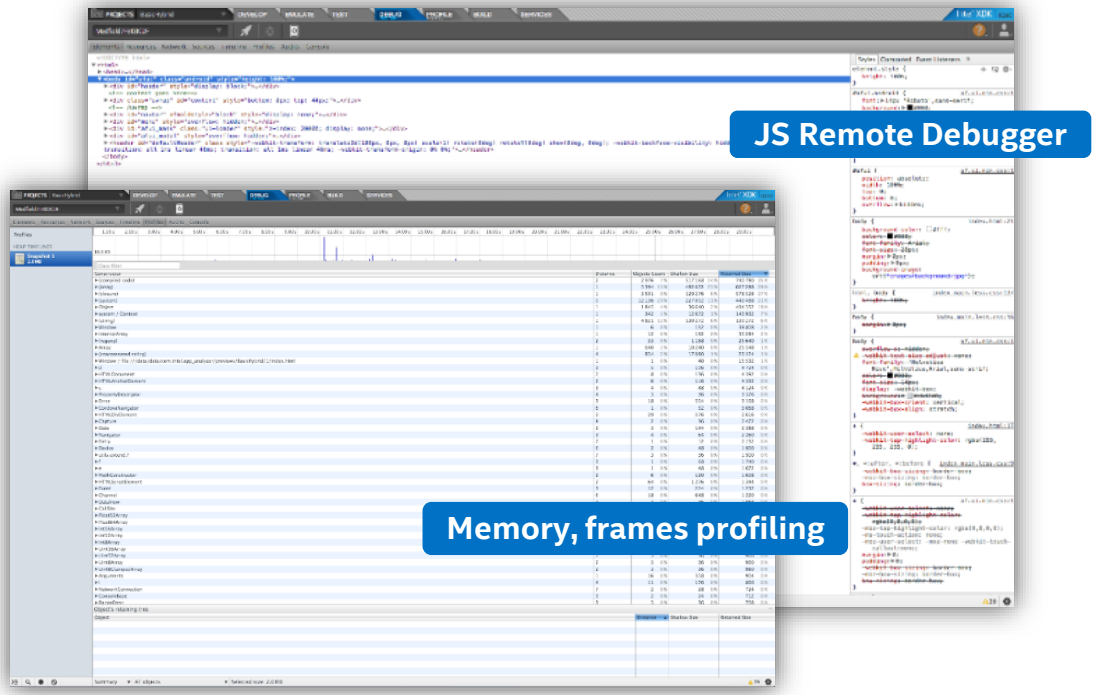
Works with Android Devices – Requires Crosswalk Run-Time

Remote on-device debugging of HTML5 code

Not building and installing required

Remote debugging provided by Google Chrome Developer Tools (CDT)

Intel App Preview (Crosswalk)



Profiler

Works with Android Devices – Requires Crosswalk Run-Time

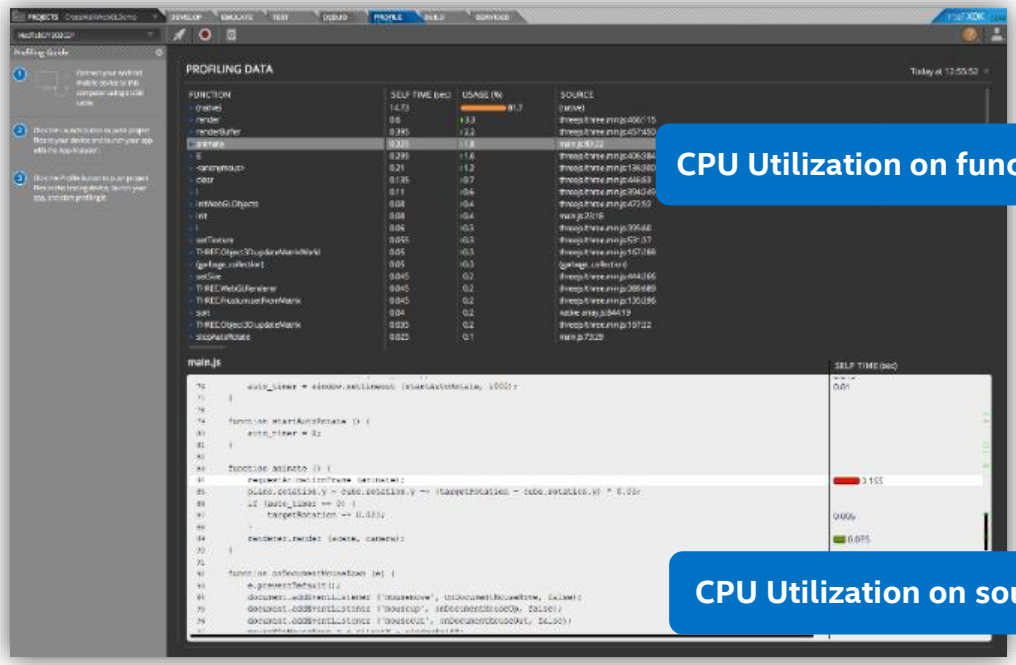
Analyze performance of JavaScript in the Android app

Collects statistics of your code

- CPU Cycles
- Source code lines

Identify hotspots

- Functions taking the most time to execute
- Time spent in each JS function



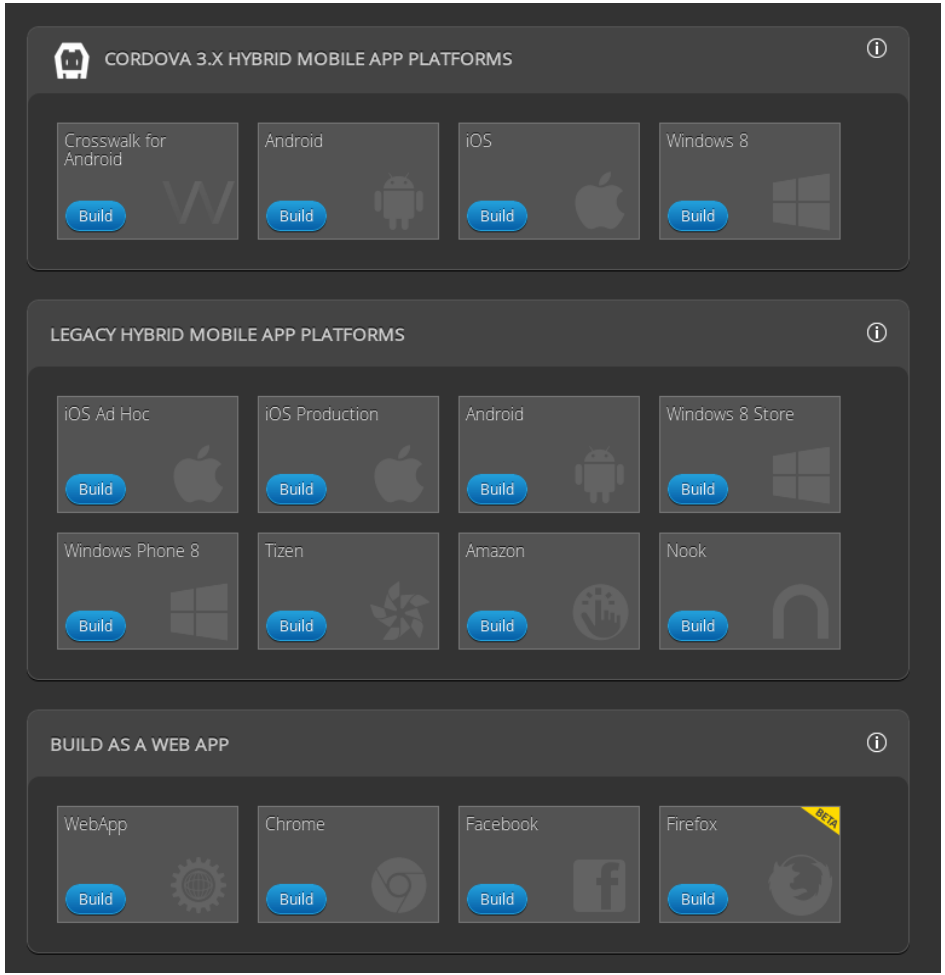
Build

HTML5 Hybrid containers:

- Standard
- Crosswalk for Android

Building on the Cloud

- Not need to set up the local environment for each supported platform
- Not hardware dependencies



Standard and Crosswalk Runtime available for Cordova 3.X building option

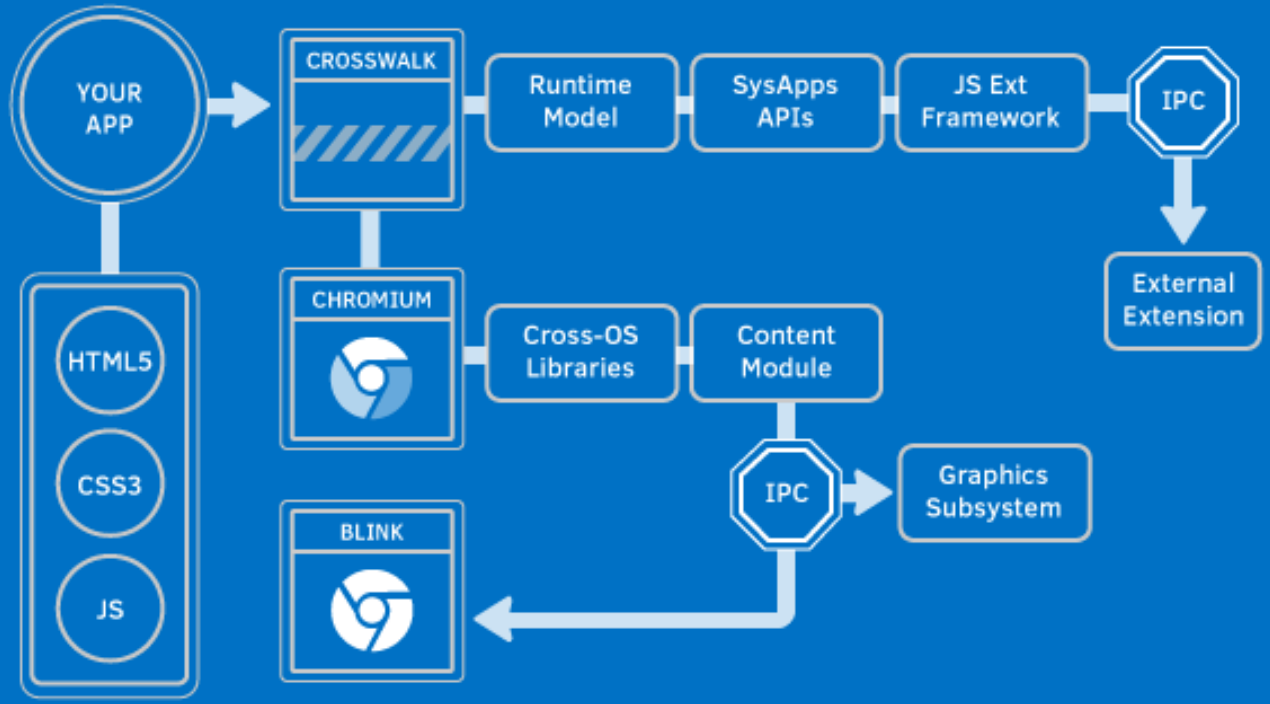
Hybrid App Packaging

Do not use Cordova 3.x plugins, but can se Cordova 2.9 APIs

Only legacy runtime available

Web App packaging

Crosswalk



Web runtime for ambitious HTML5 applications

Use experimental APIs not available in mainstream web browsers

Extends the features of a modern browser with deep device integration

API for adding native extensions

https://github.com/crosswalk-project

Crosswalk Modes

Embedded

- 2 APKs: ARM and Intel IA x86
- Bundled with the full Crosswalk runtime
- Architecture-dependent native library
- Tight dependency between Crosswalk and the app
- APK larger

Shared

- 1 APK
- Dependency with a separated Crosswalk runtime installed on the device
- Architecture-independent
- Thin layer of Java code which is architecture-independent
- 1 runtime for architecture: shared between apps
- APK smaller

```
> python make_apk.py --mode=shared --package=com.intel.xwalk-simple \
    --manifest=xwalk-simple/manifest.json
```

SIMD.JS

A set of low level APIs for programming SIMD directly in JavaScript

- The API can be mapped to the processor's SIMD instructions by a JavaScript JIT compiler when the processor has SIMD capabilities
- Default VM implementation will accomplish the task when SIMD is not available

The SIMD.JS API is architecture-neutral → Efficient SIMD execution on Intel® Architecture and ARM

Firefox Nightly

- Mozilla's Emscripten compiler modified to generate SIMD code automatically
- Major part of SIMD.JS API ready

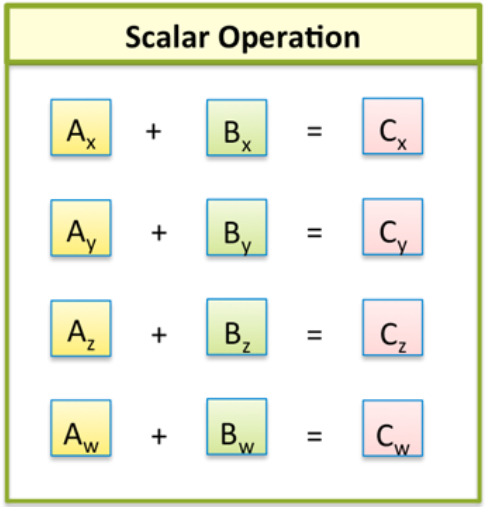
Chromium

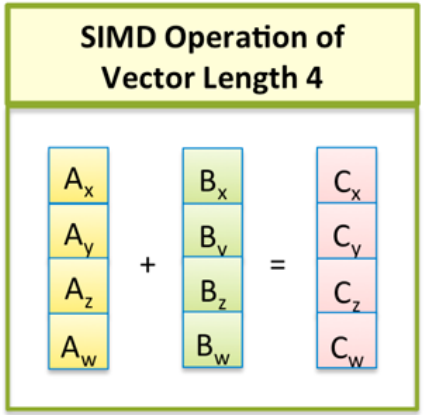
- Full implementation of the API for Intel Architecture has been submitted for review

SIMD JavaScript API: Example

A SIMD value has multiple lanes

Lanes X, Y, Z, W





Intel® Architecture currently has SIMD operations of vector length 4, 8, 16

Apply **add** operation

```
var a = SIMD.float32x4 (1.0, 2.0, 3.0, 4.0);
var b = SIMD.float32x4 (5.0, 6.0, 7.0, 8.0);
var c = SIMD.float32x4.add (a, b);
```

C = [6.0, 8.0, 10.0, 12.0] // SIMD vector as a result

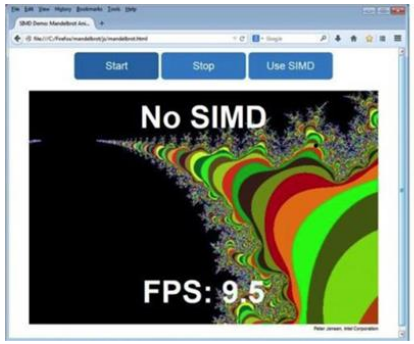
SIMD.float32x4

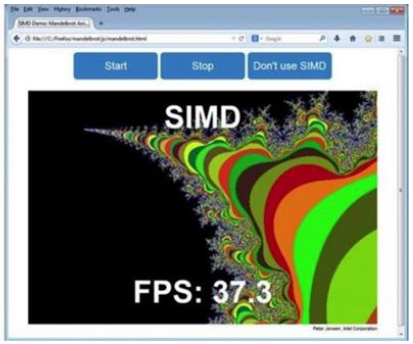
- vector with length 4
- A lane holds a IEEE-754 32-bit single-precision floating point value

Performance

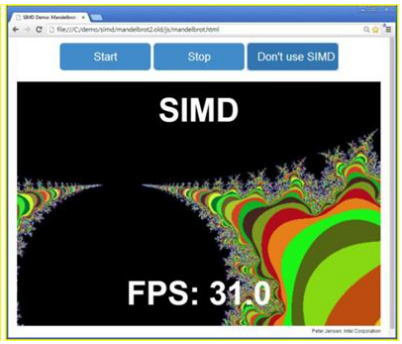
Mandelbrot set dynamically calculation as we zoom in and out

Firefox





Chrome



Crosswalk has native support for SIMD on Intel x86 architecture since version 5.34.104.0

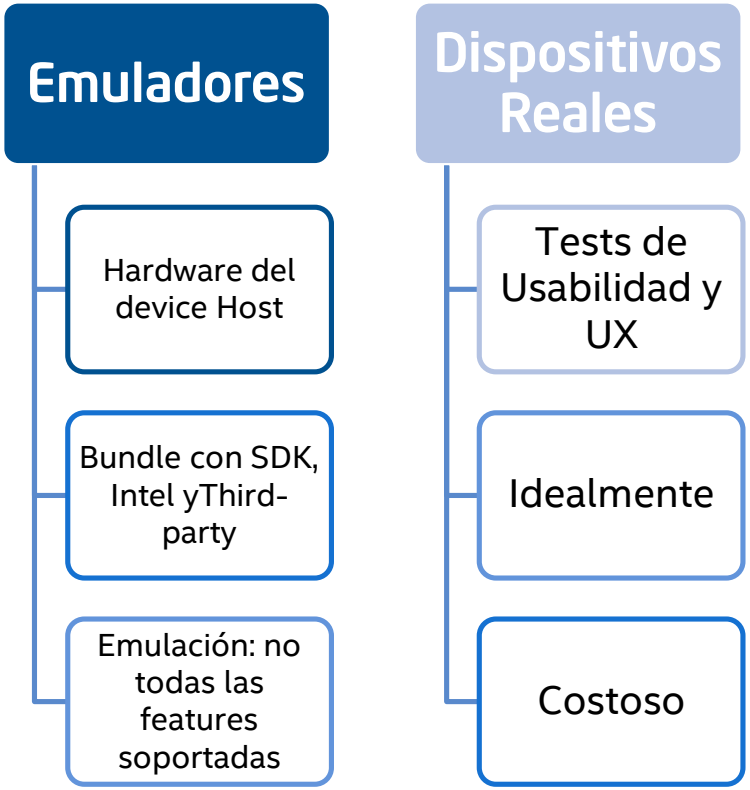
To run SIMD sample android app

- Crosswalk 5.34.104.0 version or later
- Device with an Intel x86 chipset (emulated or physical)

Cloud Testing

Testing en Múltiples Dispositivos

Android SDK
Intel HAXM
Third-party: Genymotion



Nexus, Samsung, Motorola, LG, Lenovo, y más

Cloud-based Testing

- Servicio a partir de Dispositivos Reales en el Cloud
- Tests corren en todos los dispositivos seleccionados
  - Unit Tests
  - Component Tests
  - Integration Tests
  - UI Tests
- No permite tests de Usabilidad

AppThwack

Xamarin Test Cloud

CloudMonkey Lab Manager

AppThwack

Testing de apps Android en dispositivos Intel Atom en el Cloud

- Laboratorio de dispositivos reales
- Dispositivos del mercado mundial
- No root – No jailbreak

Dispositivos incluídos:

- Asus MeMO Pad FHD 10
- Dell Venue 7
- Dell Venue 8
- Lenovo IdeaPhone K900
- Motorola Droid RAZR i
- Samsung Galaxy Tab 3 10

AppThwack

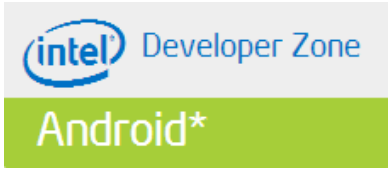
Ejecución de tests en paralelo

Recolección de datos de performance de forma automática

Frameworks soportados

- Appium
- Calabash
- Espresso
- JUnit
- KIF
- MonkeyTalk
- OCUnit
- Robotium
- Selendroid
- UI Automation
- uiautomator
- XCTest

Documentación y Links



http://software.intel.com/es-es/android

http://xdk-software.intel.com/

https://software.intel.com/en-us/html5/home

https://appthwack.com/

https://software.intel.com/es-es/articles/optimizaci-n-de-aplicaciones-android-para-arquitectura-x86

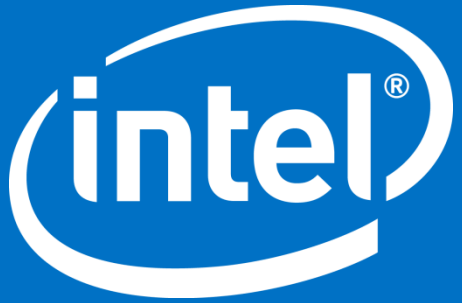
http://software.intel.com/es-es/articles/android-application-development-and-optimization-on-the-intel-atom-platform

http://software.intel.com/en-us/blogs/2012/12/12/from-arm-neon-to-intel-mmxsse-automatic-porting-solution-tips-and-tricks

http://software.intel.com/en-us/android/articles/ndk-android-application-porting-methodologies

http://software.intel.com/en-us/blogs/2014/03/19/free-ebook-download-from-apress-android-on-x86-an-introduction-to-optimizing-for

http://ark.intel.com/es/Products/VirtualizationTechnology



Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804