



Performance of Cross-Platform Mobile Applications

Michiel Willocx
MSEC
3 feb 2016



Table of content

- Introduction
- Test strategy
- Overview of the results

JavaScript frameworks

Runtimes

Source code translators

Web-to-native wrappers



Intel's App Framework



Sencha



ICENIUM™



PhoneGap



ENYO



Corona SDK



ADOBE AIR™



AppGyver®

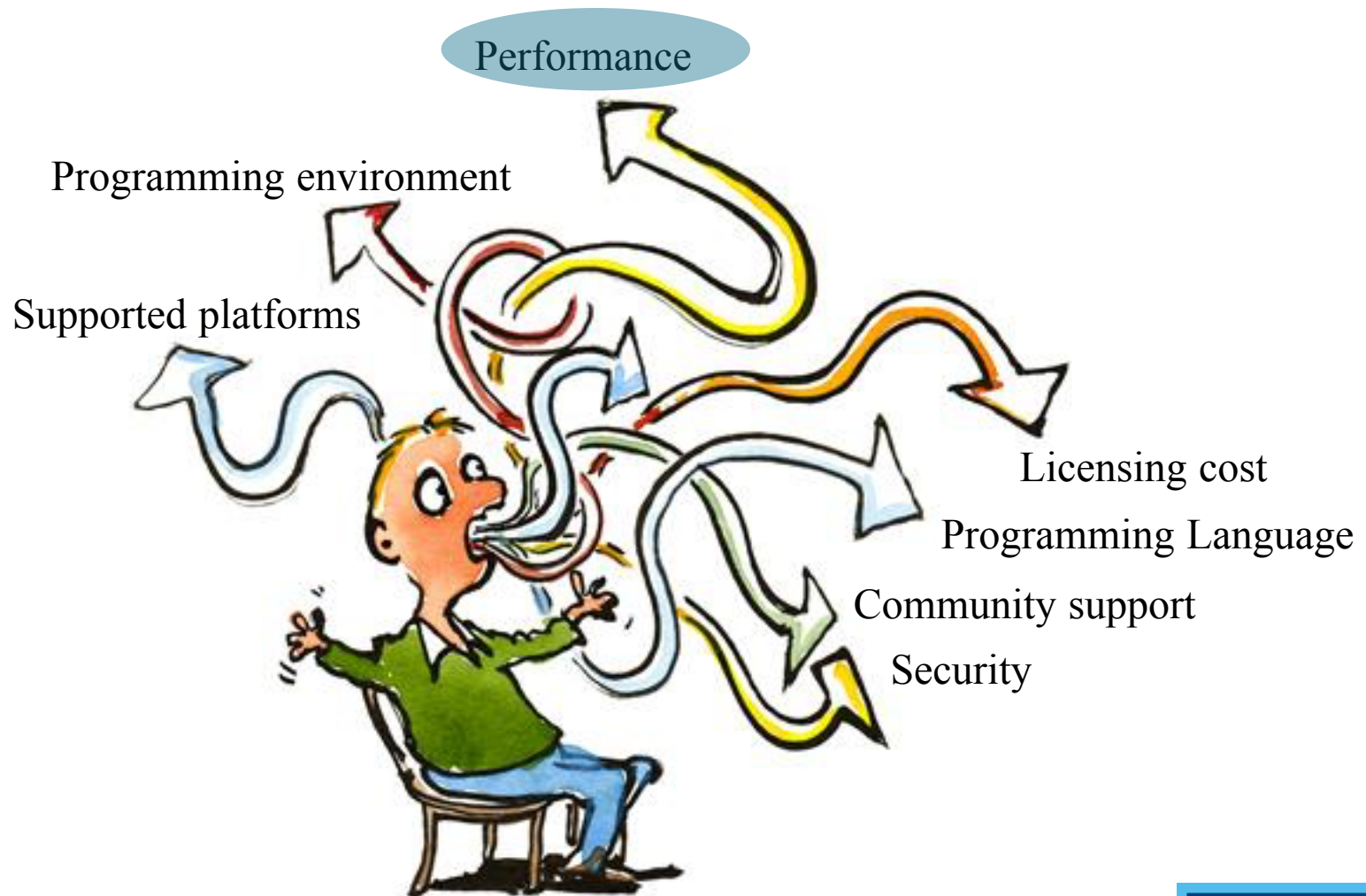
NEOMAD



unity



Cross-platform tool selection



Overview Performance Analysis

Overall application performance Overall application behaviour

- Response times
- CPU usage
- Memory Usage
- Battery usage
- Disk Space

Sensor access performance Hardware access performance Native API access performance

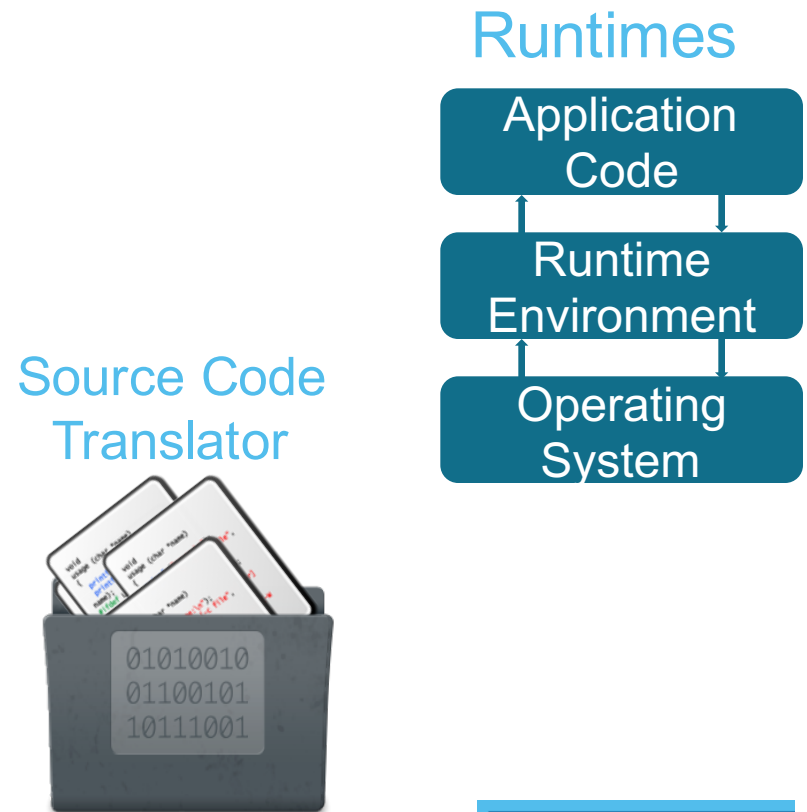
- GPS
- NFC
- ...
- Graphical Performance
- ...
- Local Storage
- Address book access
- ...

Cross-platform technology

1) Based on web technology



2) Not based on Web technology



Web Apps

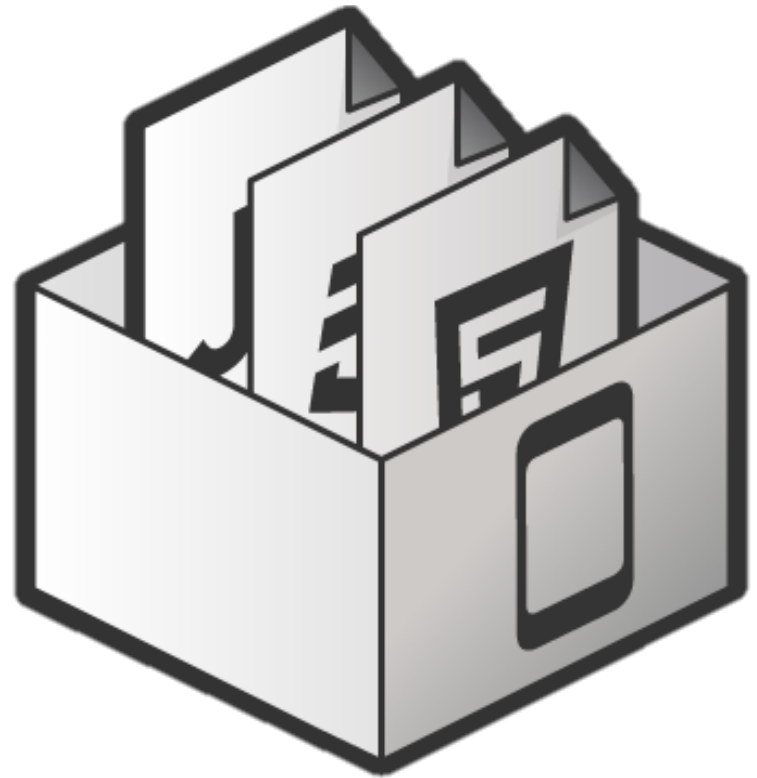


→ Mobile Websites

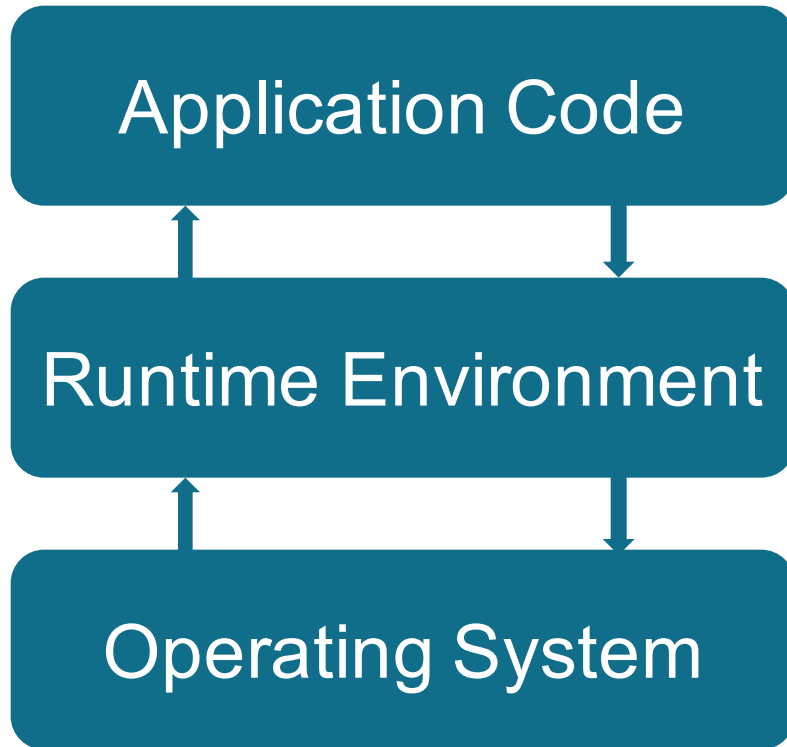
- Accessed in standard mobile browser (Chrome, Safari, ...)
- Optimized for mobile device screen sizes
- Use **JavaScript Frameworks**
 - UI Components
 - Event handling, utility functions
 - Use of design patterns

Web-To-Native Wrappers

- Web Apps, packaged as a stand-alone application
- Web code is displayed in a chromeless webview
- Wider range of native API calls compared to normal Web browser



Runtimes



- Cross-platform compatibility layer
- Shields app from underlying differences between platforms
- Different strategies:
 - Interpreted at runtime
 - Compiled in advance (*source code translators*)

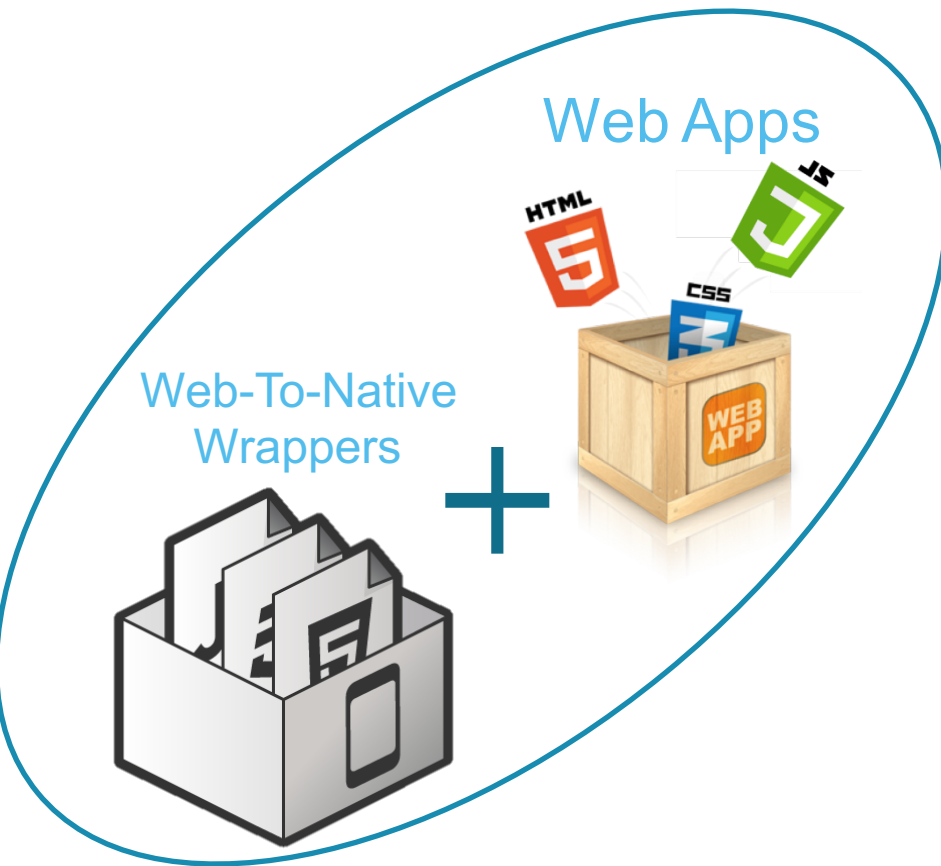
Source Code Translator



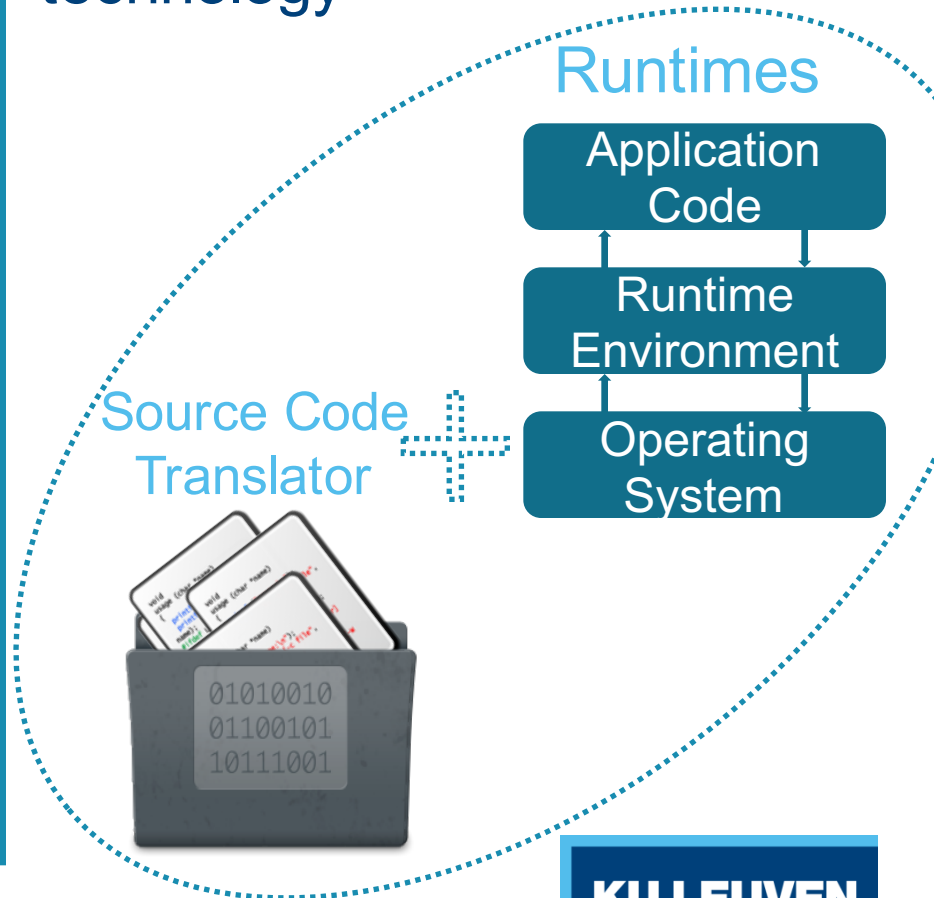
- Cross-compilation of code
- Different strategies:
 - Translate to native source
 - Translate to intermediary language
 - Translate to low level machine code
- Often used in combination with Runtime

Cross-platform technology

1) Based on web technology



2) Not based on Web technology



Test Strategy



PropertyCross

Helping you **select** a cross-platform mobile framework

[Download \(v1.6\)](#)[View on GitHub](#)[Blog](#)

Introduction

Developers are now finding themselves having to author applications for a diverse range of mobile platforms (iOS, Android, Windows Phone, ...), each of which have their own 'native' development languages, tools and environment.

There is an ever growing list of cross-platform frameworks that allow you to minimise the cost and effort of developing mobile apps, but which to choose?

To help solve this problem PropertyCross presents a non-trivial application, for searching UK property listings, developed using a range of cross-platform technologies and frameworks. Our aim is to provide developers with a practical insight into the strengths and weaknesses of each framework.

This project is part of **TasteJS**, which also includes **TodoMVC** - a project that helps developers compare JavaScript frameworks.

New in v1.6 - July 22nd, 2014

- **Famous** implementation added.
- **Intel App Framework** implementations updated to improve UI.
- **Native** implementation updated to take advantage of features in Xcode 5.

New in v1.5 - May 7th, 2014











































- **Lavaca** implementation added.
- **Enyo** implementation updated.
- **Sencha Touch 2** implementation updated to use **Sencha Touch 2.3 new themes**.
- **Titanium**, **Xamarin**, **Native** and **Kendo UI** implementations updated with iOS 7 look and feel.
- **Kendo UI** implementation updated to support Windows Phone 8.

New in v1.4 - March 24th, 2014

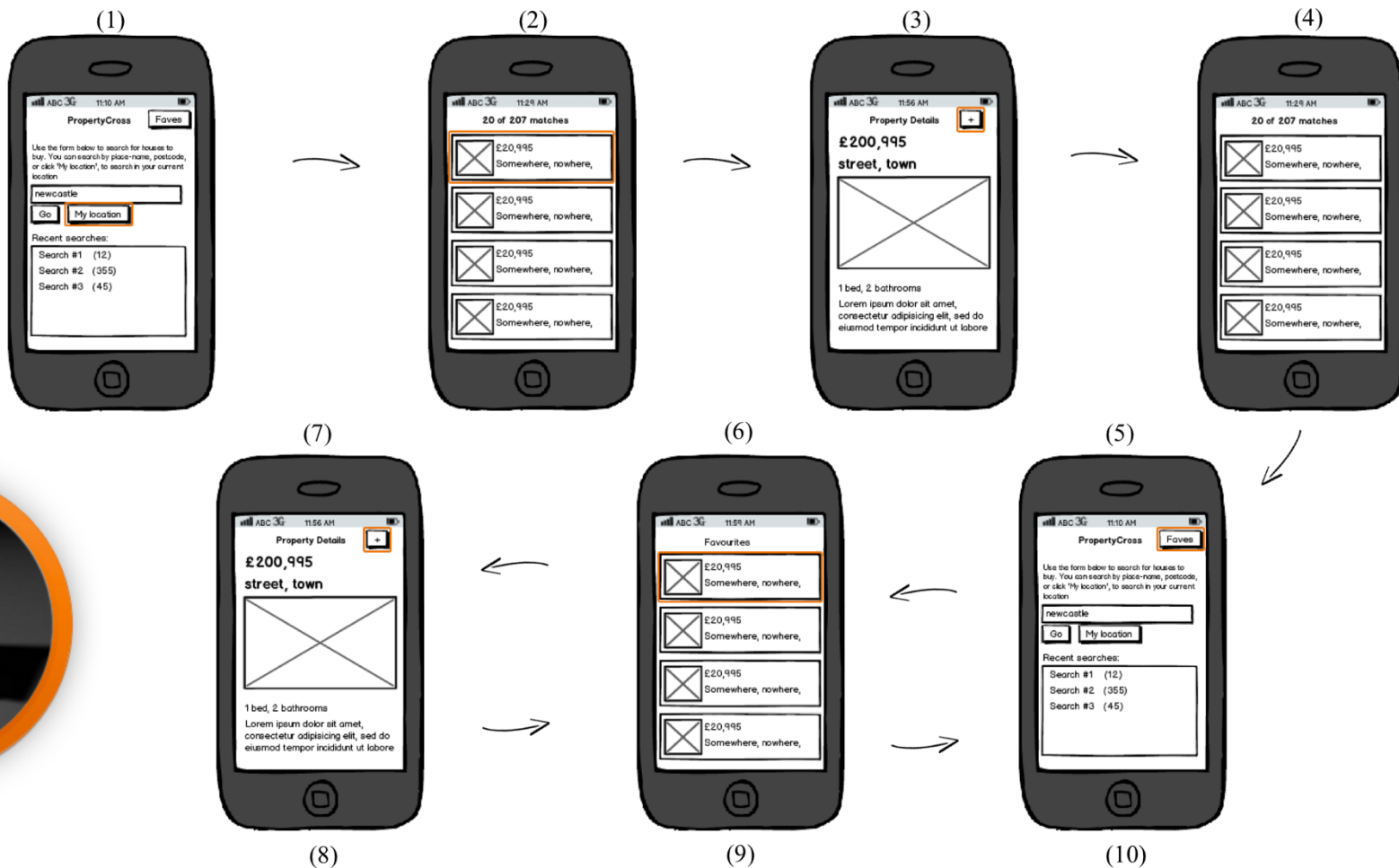
- **Ionic** implementation added.
- **Native** implementation updated.



Frameworks

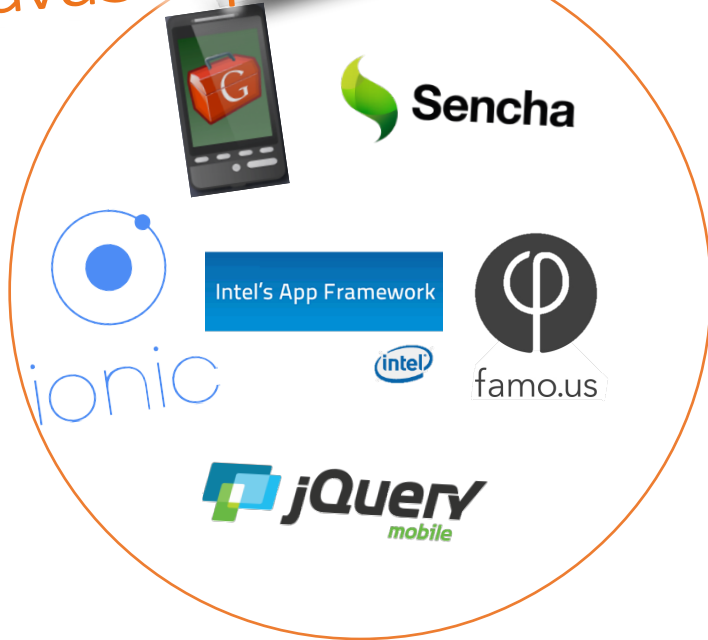
	AIR			n/a
	Delphi			n/a
	Emy *			
	Enyo *			
	Famo.us *			
	Intel App Framework *			
	Ionic *			n/a
	jqTouch *			n/a
	jQuery Mobile *			
	Kendo UI *			
	lavaca *			n/a
	AngularJS *			n/a

WIREFRAME

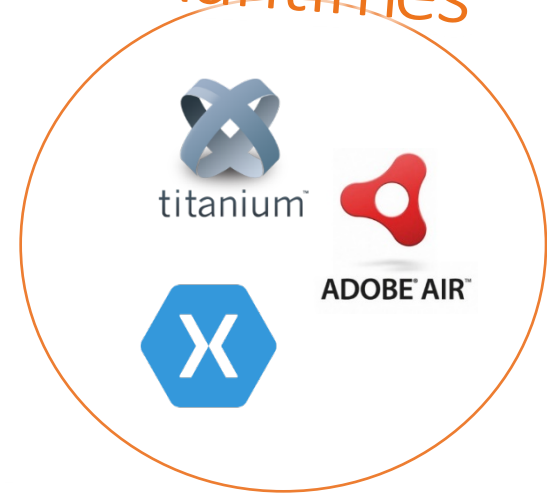


Tested cross-platform tools

JavaScript Frameworks



Runtimes



Native

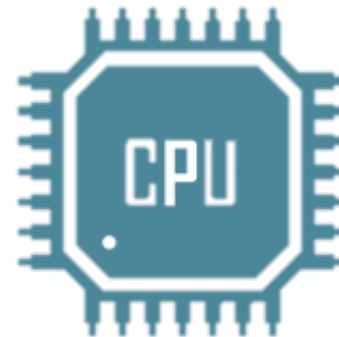
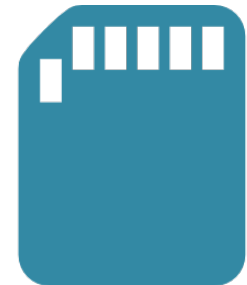
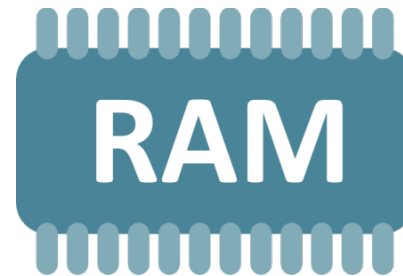


Source Code Translator



Measured Parameters

- Start time
- In-app response times
- Memory usage
- CPU usage
- Disk space



Scope & Approach: Devices

	Low-end	High-end
iOS		
Device:	iPhone 4	iPhone 6
OS:	iOS 7.1.2	iOS 9.1
RAM:	512 MB	1 GB
CPU:	1 GHz	Dual-core, 1.4 GHz
		
Android		
Device:	Sony Xperia E3	Motorola Nexus 6
OS:	Android 4.4.2 (KitKat)	Android 6.0 (Marshmallow)
RAM:	1 GB	3 GB
CPU:	Quad-core, 1.2 GHz	Quad-core, 2.7 GHz
		
Windows Phone		
Device:	Nokia Lumia 925	
OS:	Windows 8.1	
RAM:	1 GB	
CPU:	Dual-core, 1.5 GHz	
		

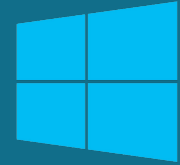
Measuring tools



Android



iOS



Windows Phone

	Android	iOS	Windows Phone
Response times	DDMS	Instruments Tool (Time Profiler)	Visual Studio Console
CPU usage	ADB “top”	Instruments Tool (CPU Activity)	Windows Phone Developer Power Tools
Memory usage	ADB “dumpsys meminfo”	Instruments Tool (Allocations)	Windows Phone Developer Power Tools
Disk space	Visible on device	Visible on device	Visible on device

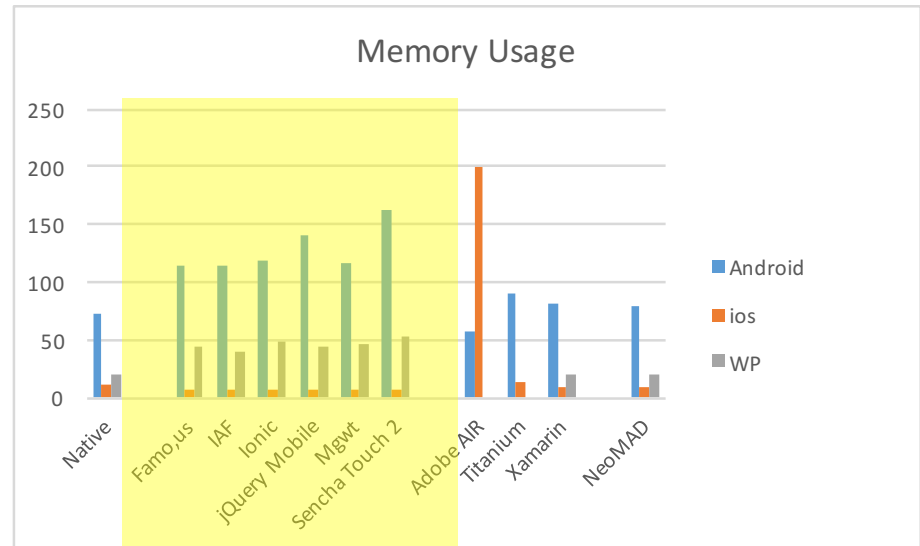
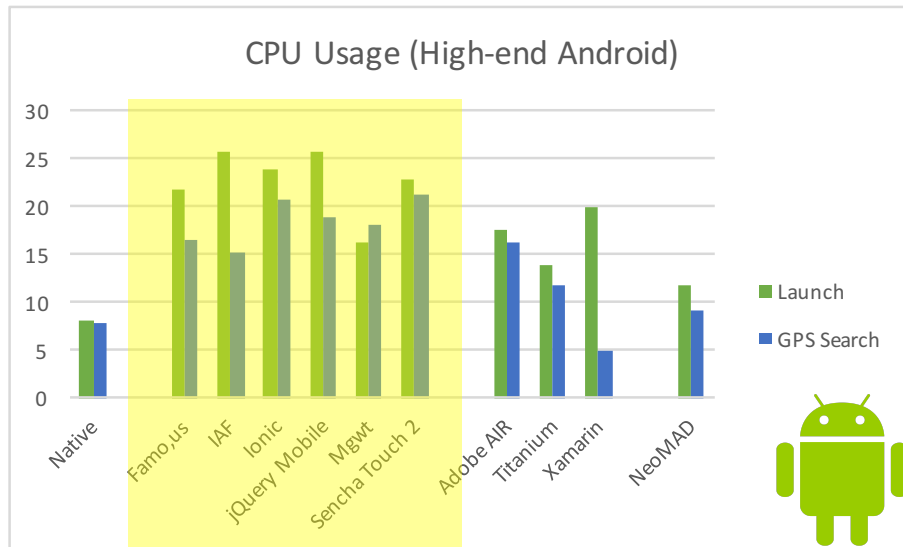
Results



Cross-platform tools of the same category show similar behavior

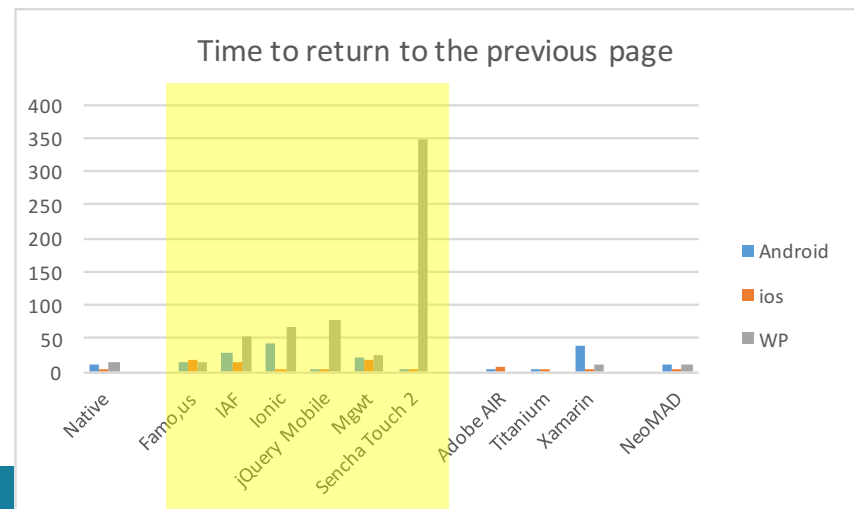
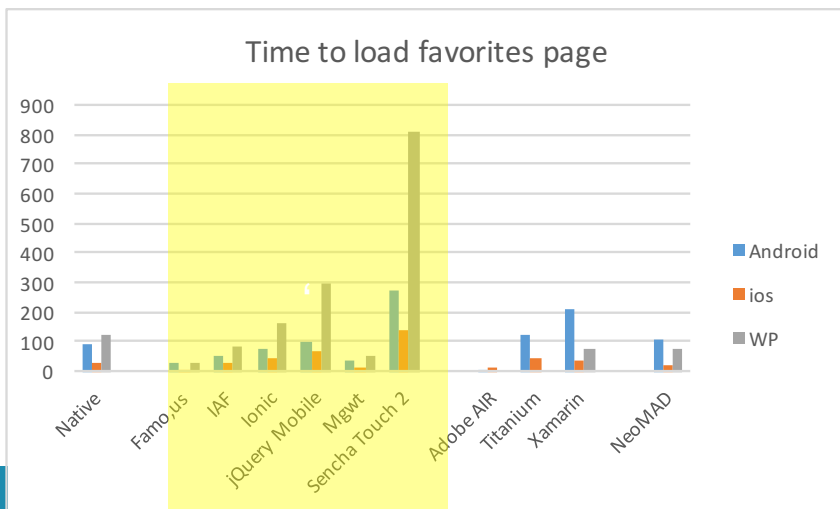
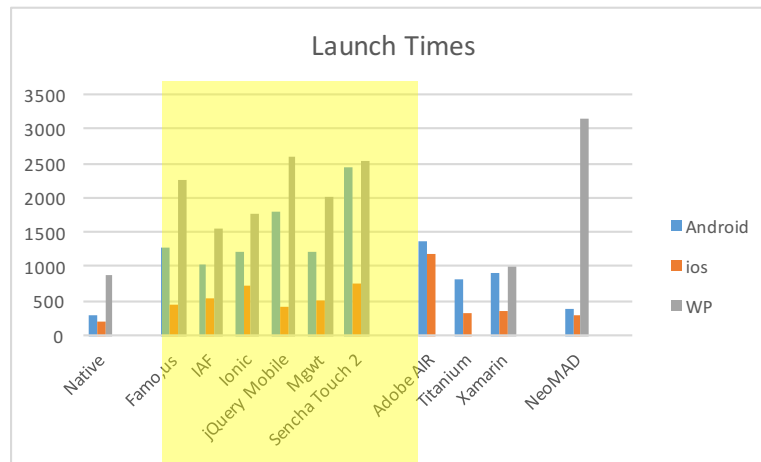
Cross-platform tools of the same category show similar behavior

JavaScript Frameworks



Cross-platform tools of the same category show similar behavior

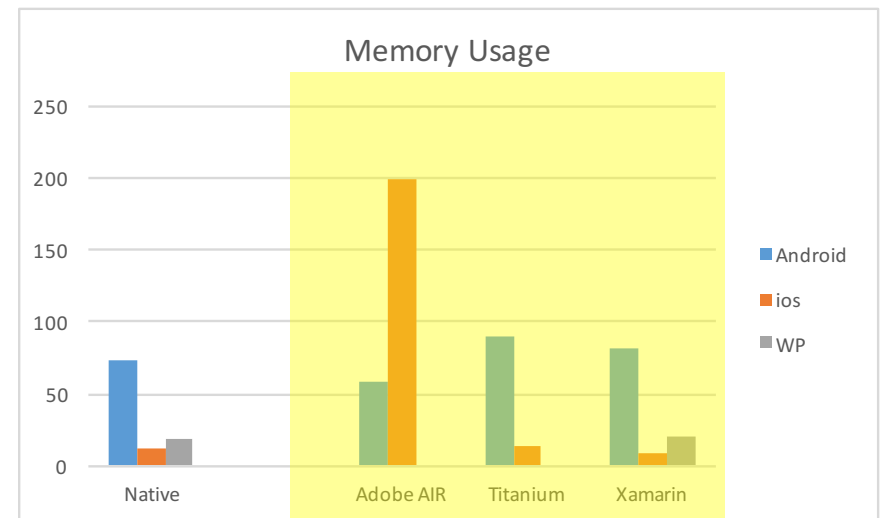
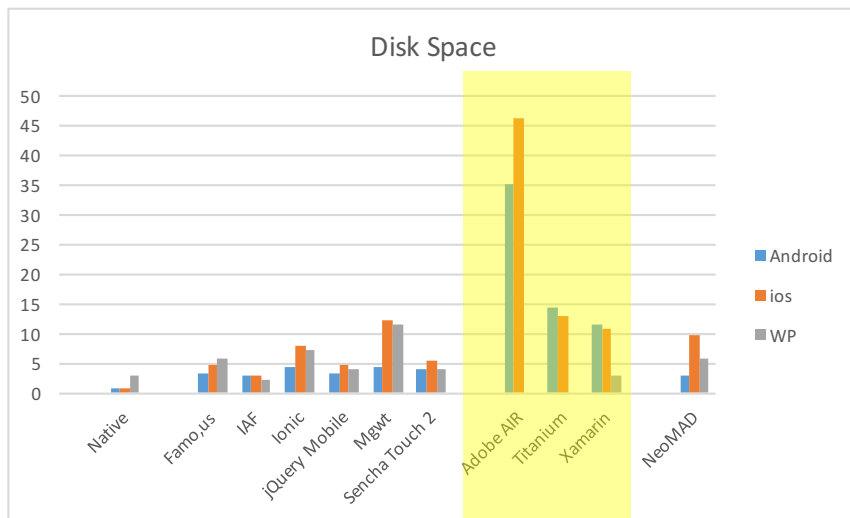
JavaScript Frameworks



Displayed values are measured on the high-end devices

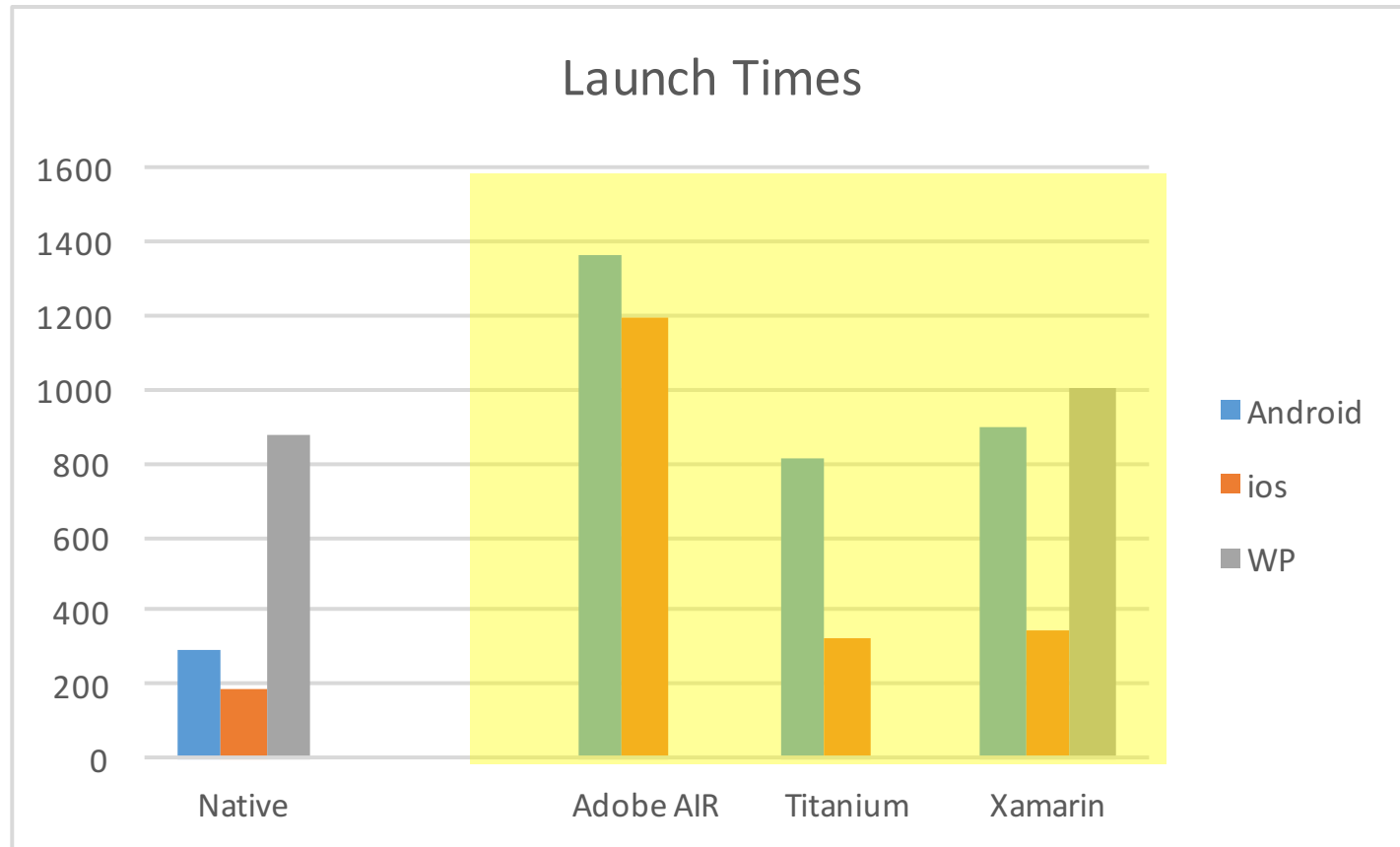
Cross-platform tools of the same category show similar behavior

Runtimes



Cross-platform tools of the same category show similar behavior

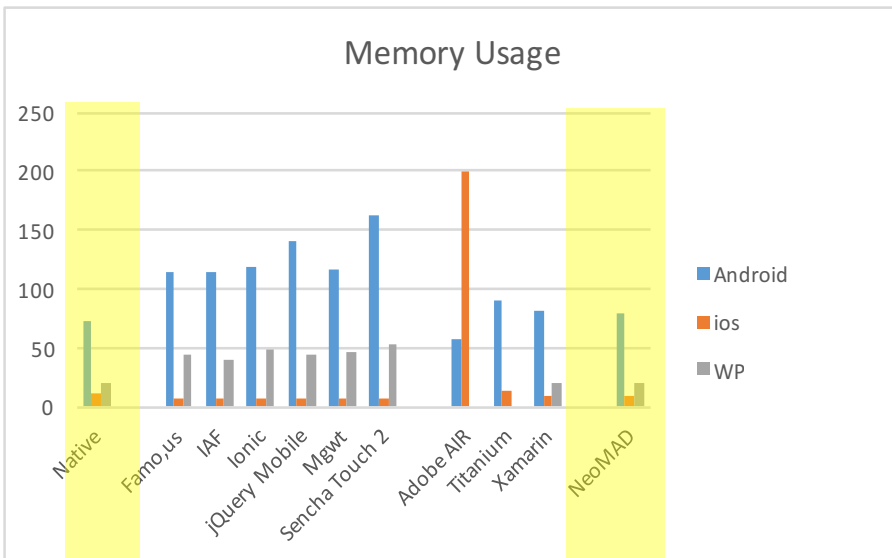
Runtimes



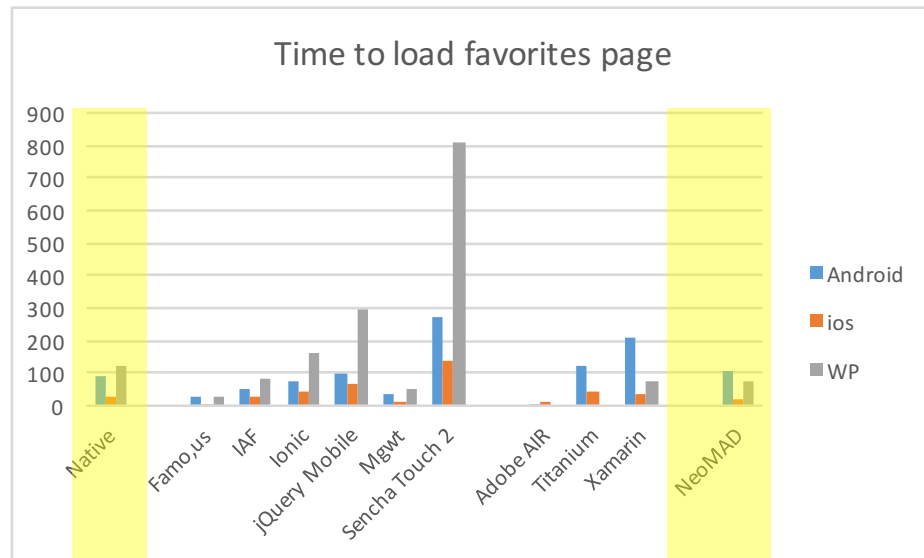
Cross-platform tools of the same category show similar behavior

Source Code Translator

Memory Usage



Time to load favorites page

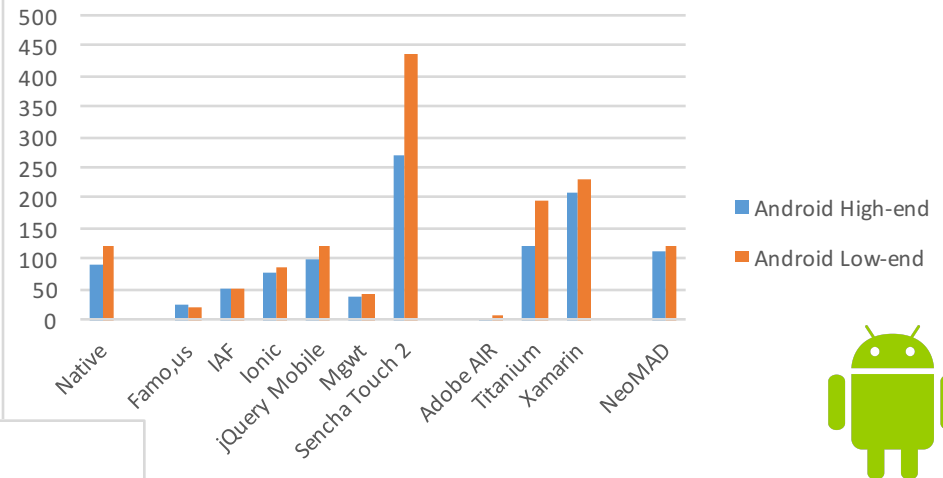


The performance penalty resulting from the use of cross-platform tools is generally acceptable

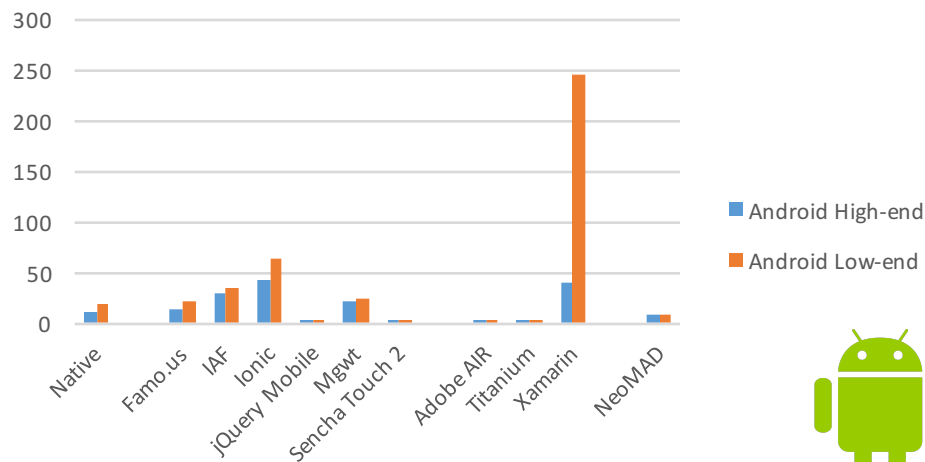
The performance penalty resulting from the use of cross-platform tools is generally acceptable

Vb1: In app response times

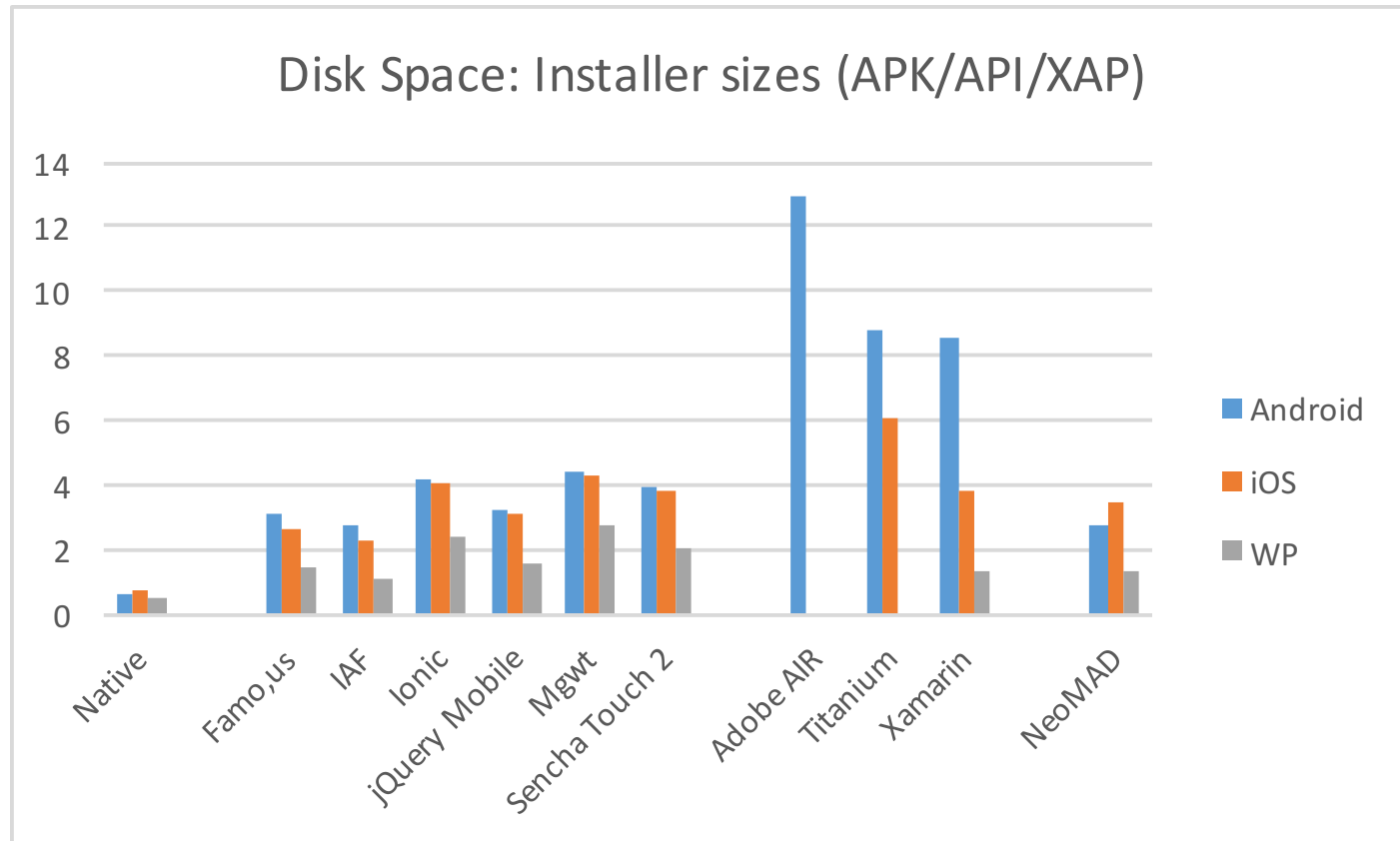
Time to open a favorite page of the application



Time to return to previous page of the app

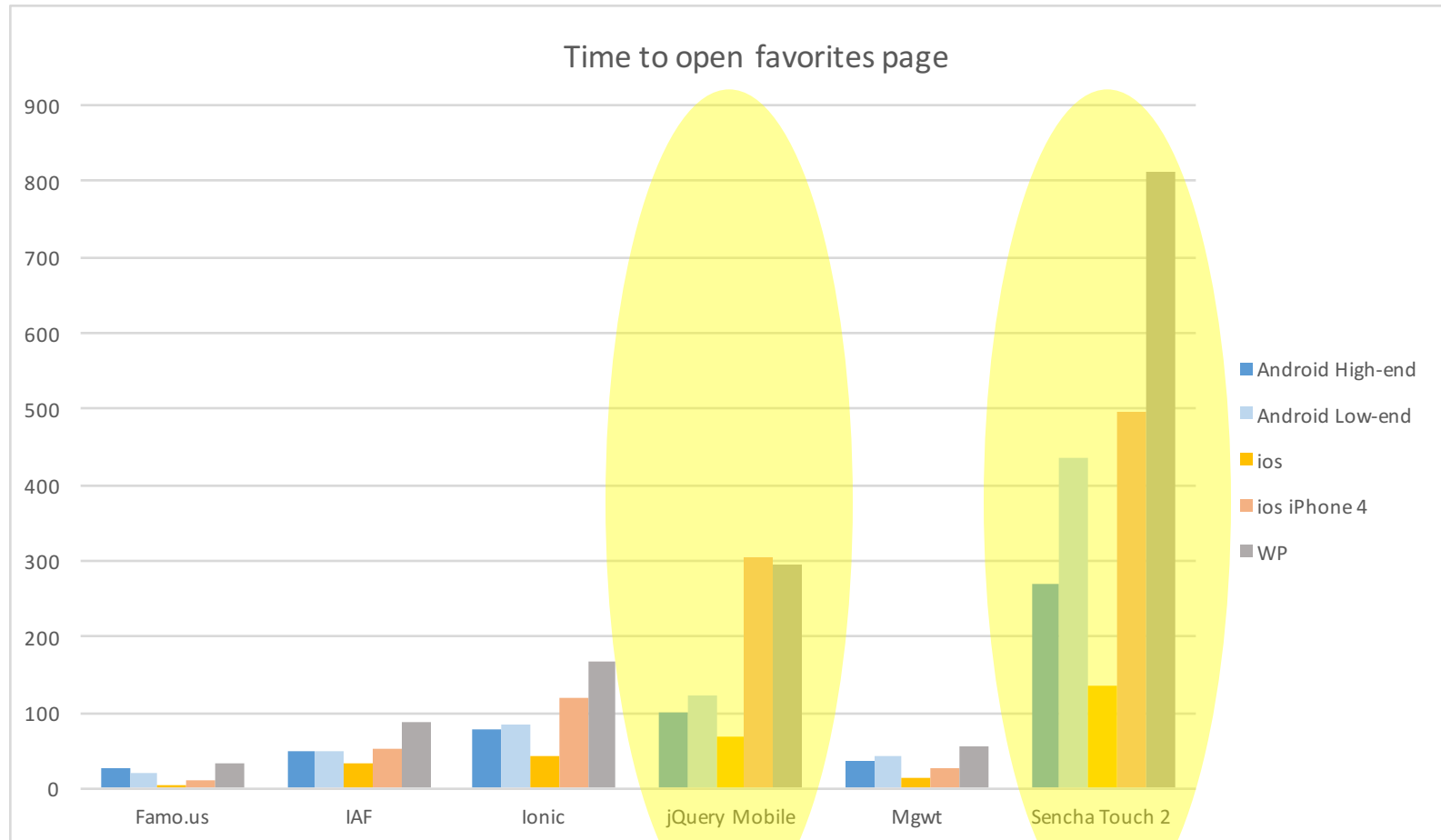


The performance penalty resulting from the use of cross-platform tools is generally acceptable



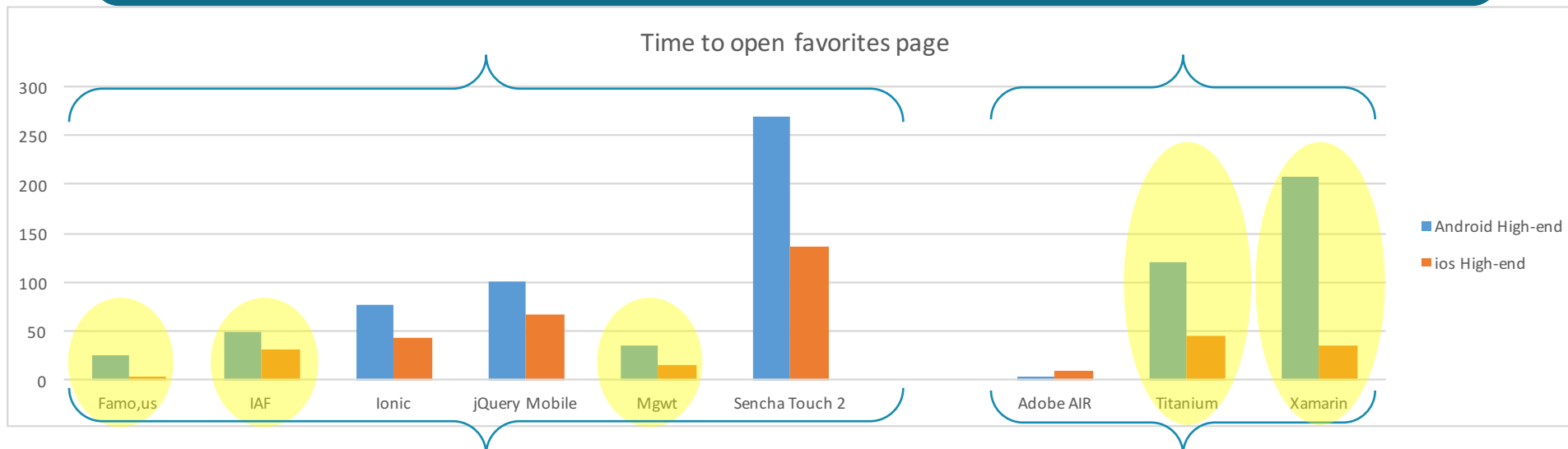
Underlying differences between different JavaScript frameworks have an effect on the performance

Underlying differences between different JavaScript frameworks have an effect on the performance



Page rendering: JavaScript frameworks vs Runtimes, speed vs Native UI components

Page rendering: JavaScript frameworks vs Runtimes, speed vs Native UI components



JavaScript Frameworks

- Webview renders HTML pages
- Some JavaScript frameworks have faster response times than native apps
- Sometimes native skins
No real, native UI components

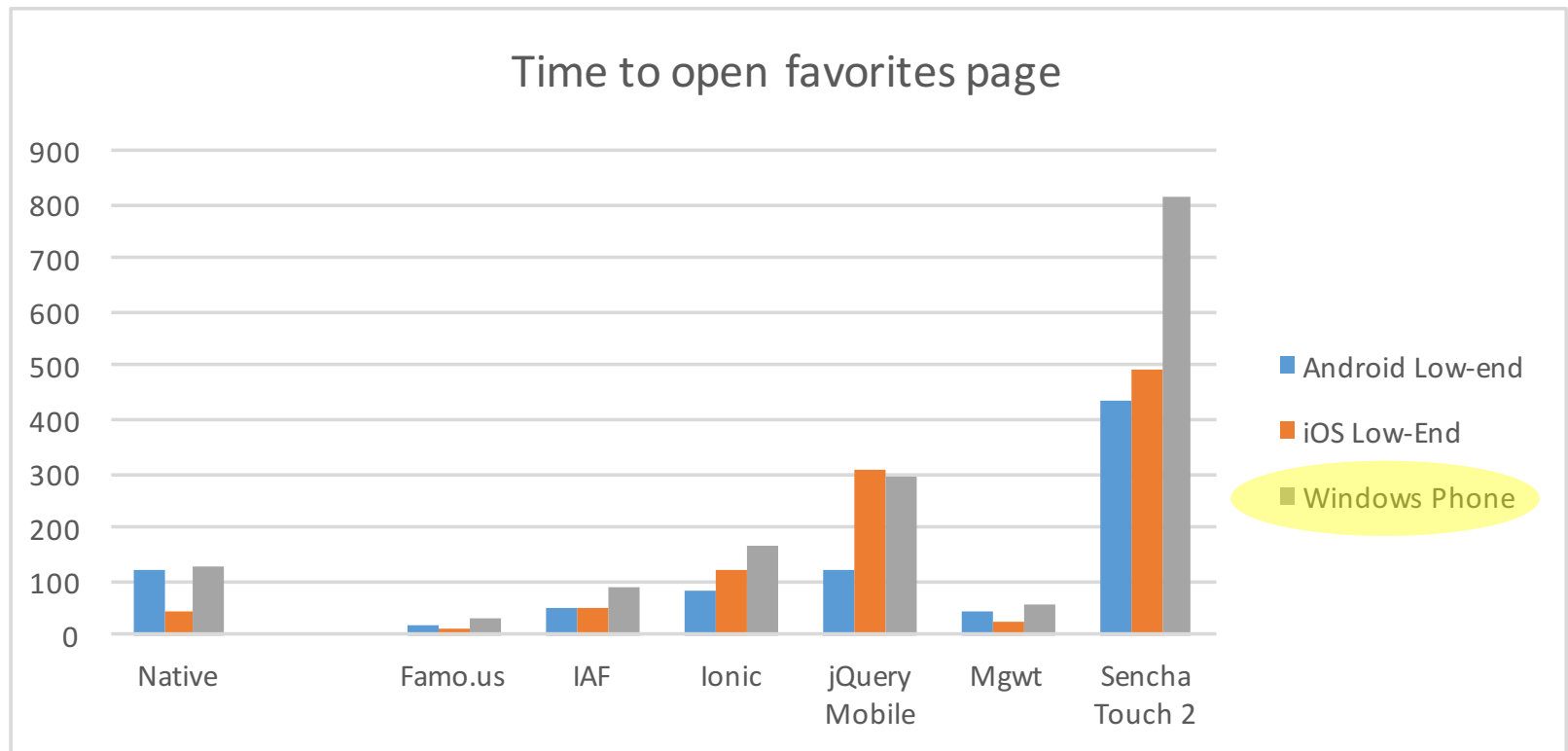
Runtimes

- Creates UI View elements
- Makes use of Native UI components
- Additional overhead introduced

**The performance of a cross-platform application
strongly depends on the targeted platform**

The performance of a cross-platform application strongly depends on the targeted platform

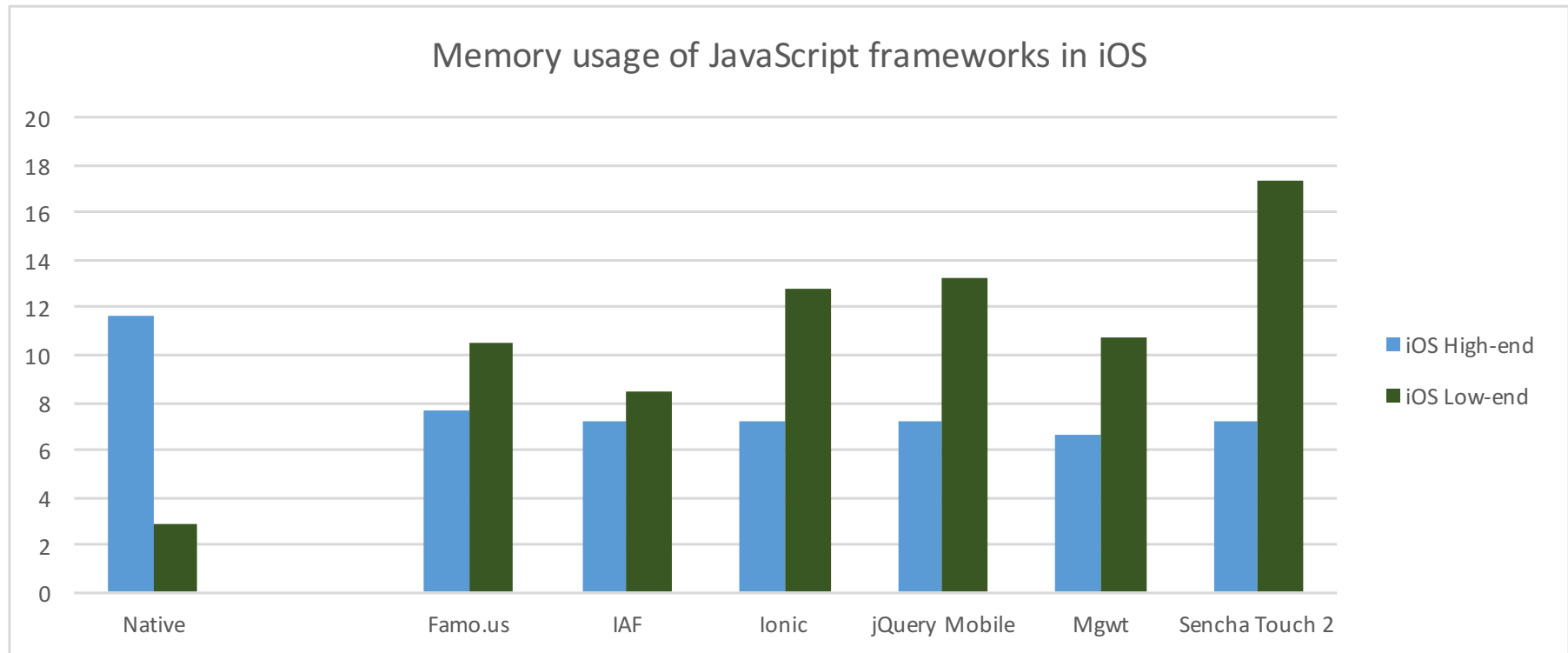
The Webview/JavaScript Engine



→ between different platforms

The performance of a cross-platform application strongly depends on the targeted platform

The Webview/JavaScript Engine



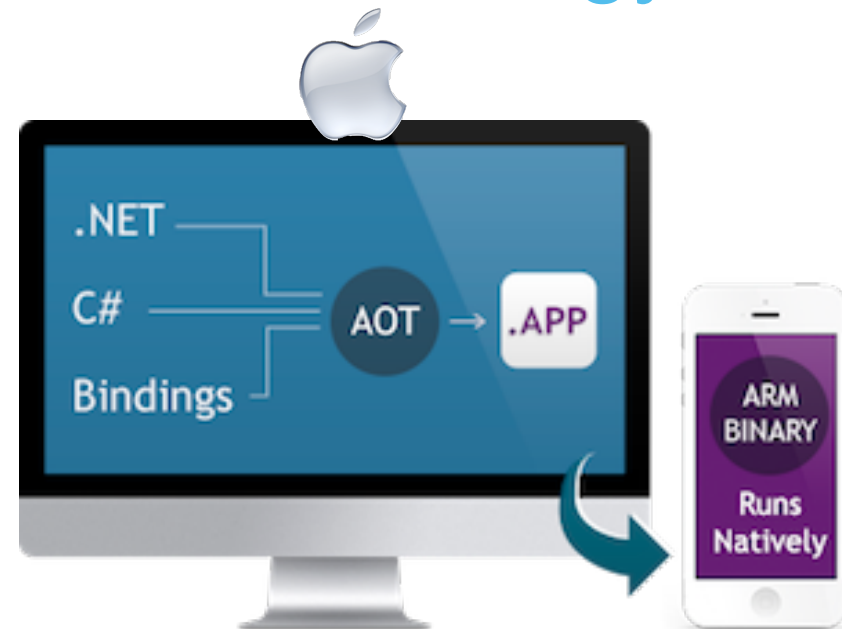
→ between different versions of the same platform

The performance of a cross-platform application strongly depends on the targeted platform

Xamarin: Same tool, different strategy



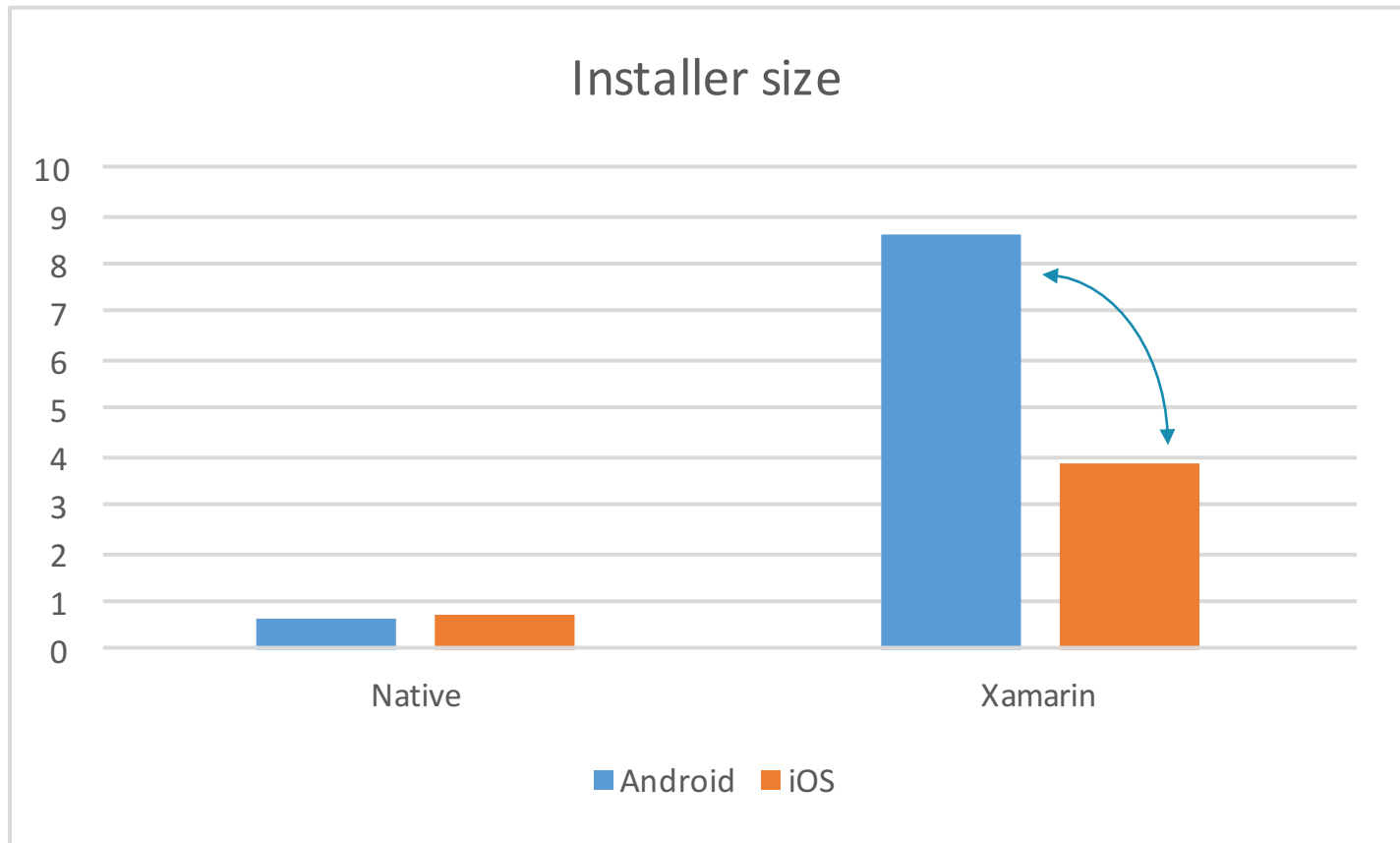
- Source translated to Intermediary Language (IL)
- Just-In-Time (JIT) compilation



- Source translated to executable binary code
- Ahead-Of-Time (AOT) compilation

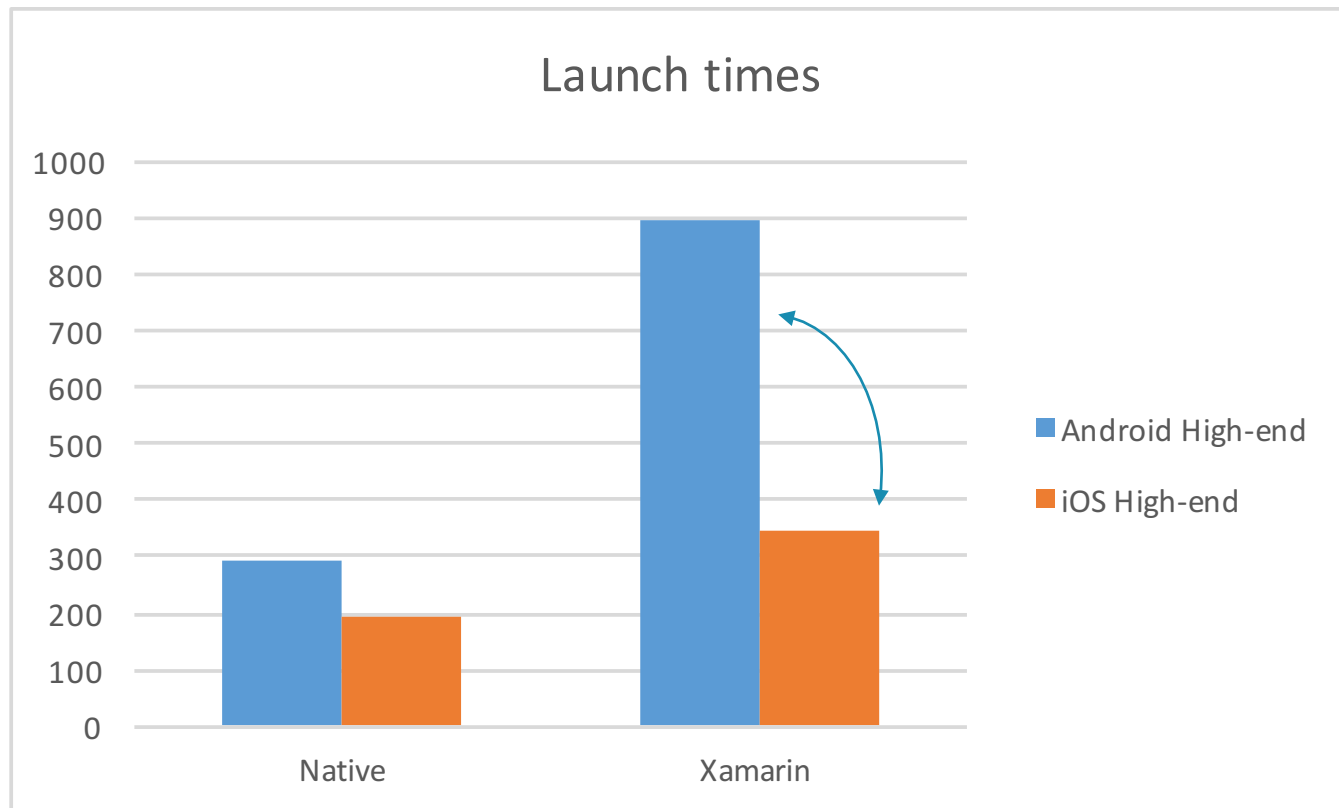
The performance of a cross-platform application strongly depends on the targeted platform

Xamarin: Same tool, different strategy



The performance of a cross-platform application strongly depends on the targeted platform

Xamarin: Same tool, different strategy



The performance of a cross-platform application strongly depends on the targeted platform

Android: other version, different strategy

Dalvik

Used in all Android versions until Android 4.4

Just-In-Time (JIT) compilation

For each run of the app, the part of the code required for its execution is compiled to machine code at that moment

ART

Officially introduced with Android 5.0

Ahead-Of-Time (AOT) compilation

The whole application is pre-compiled (only once) at install time into a system-dependent binary

Consequences on the performance

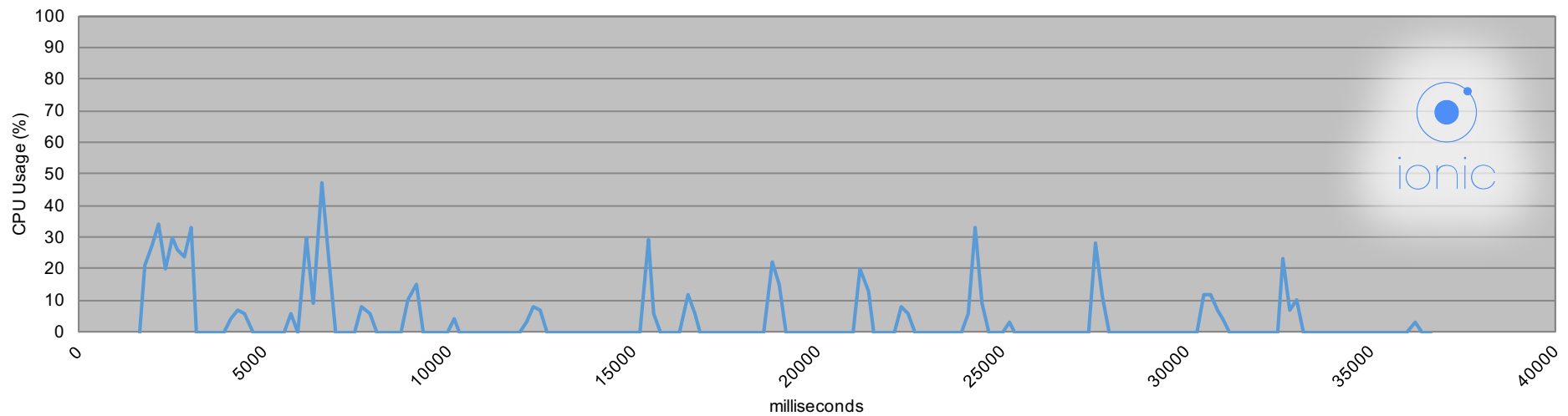
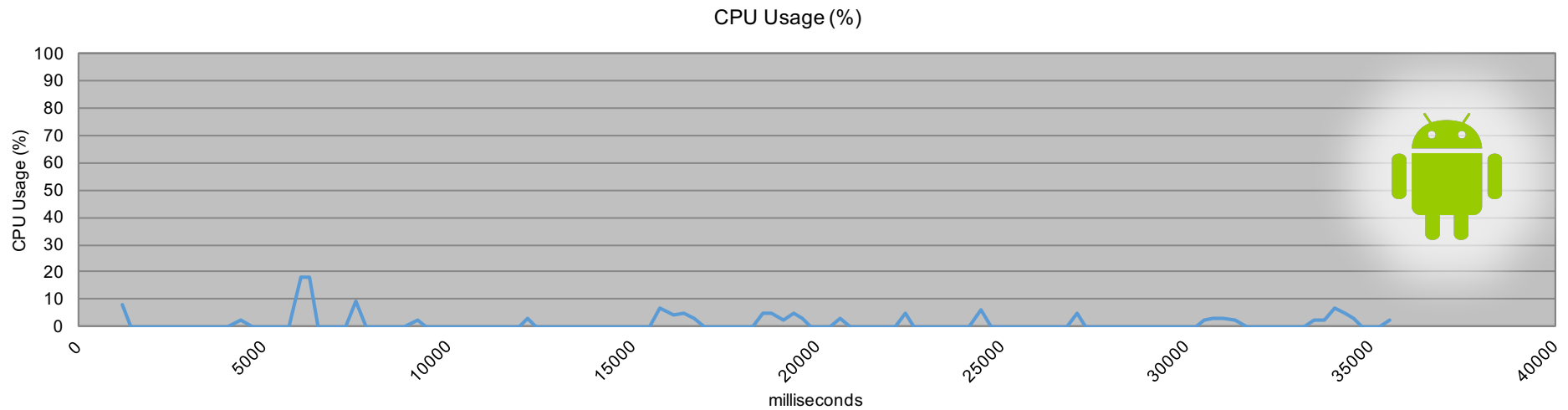
- More CPU intensive
- Uses more battery
- Slower

- More disk space required
- Installation process more time consuming
- Larger memory footprint

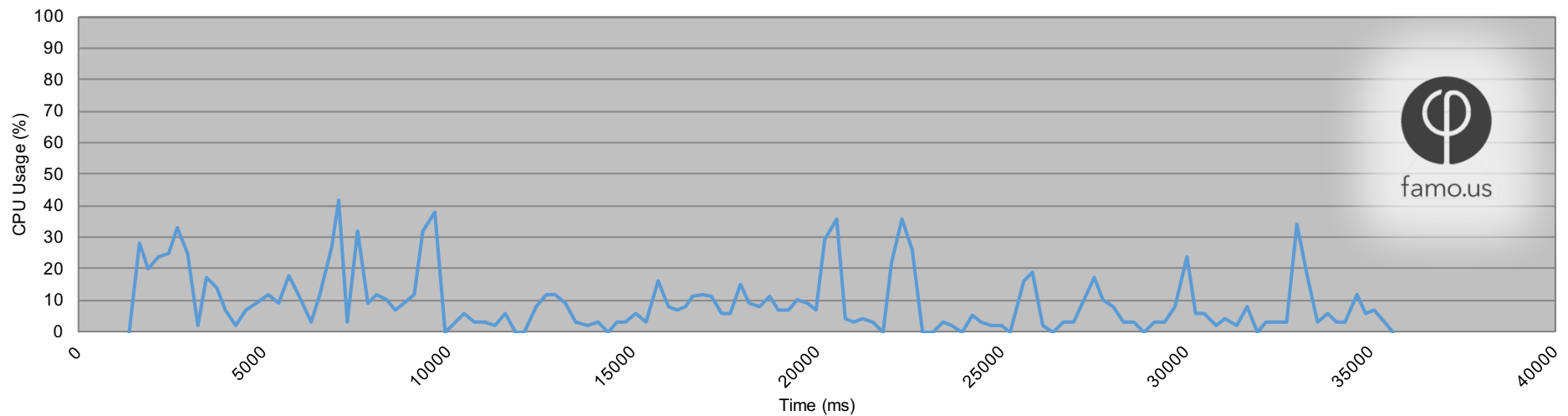
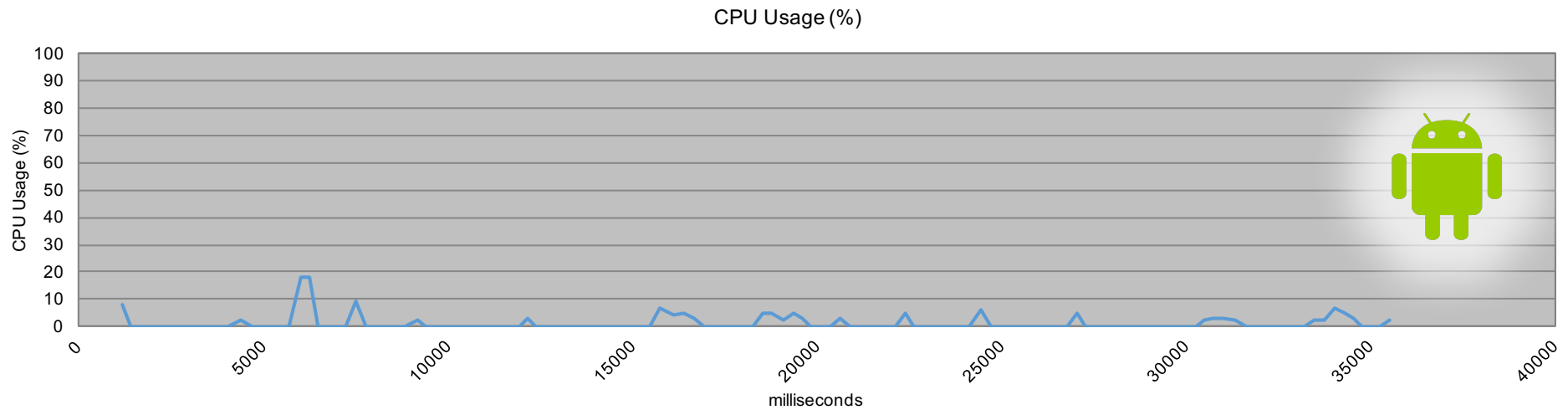
A closer look at some parameters



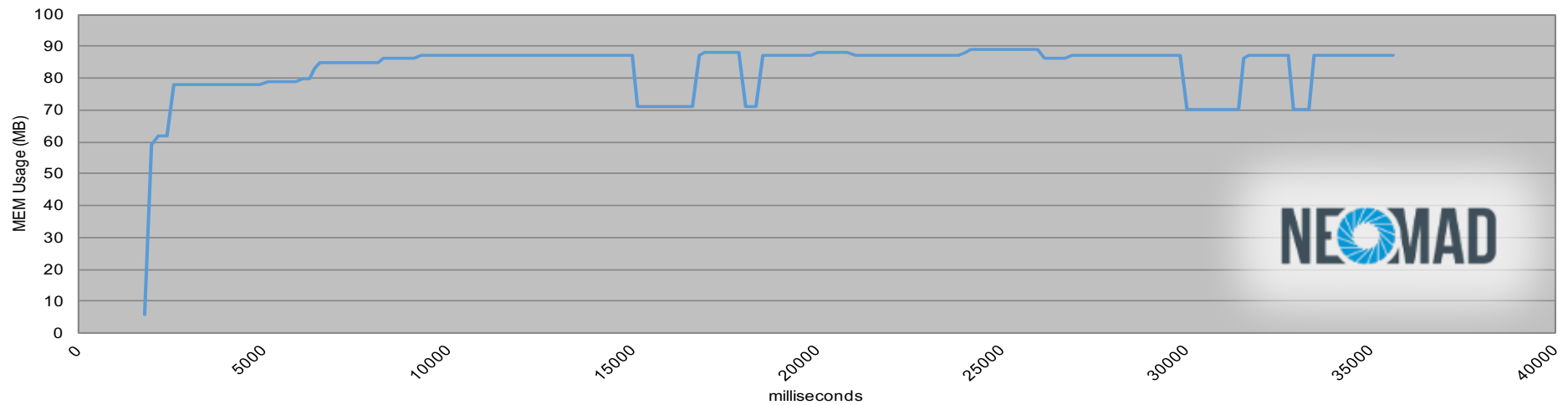
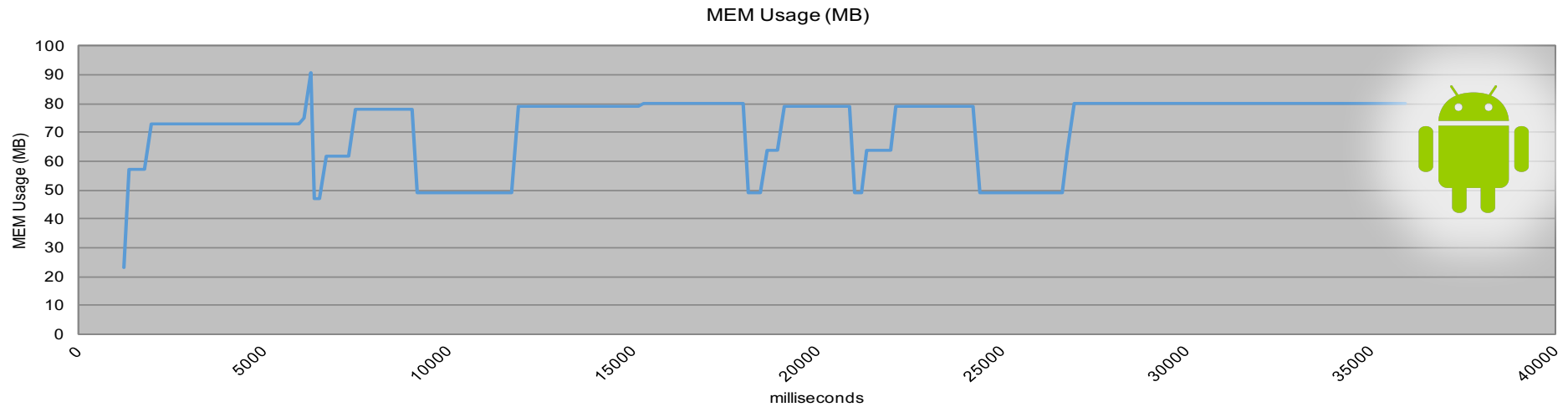
CPU Usage



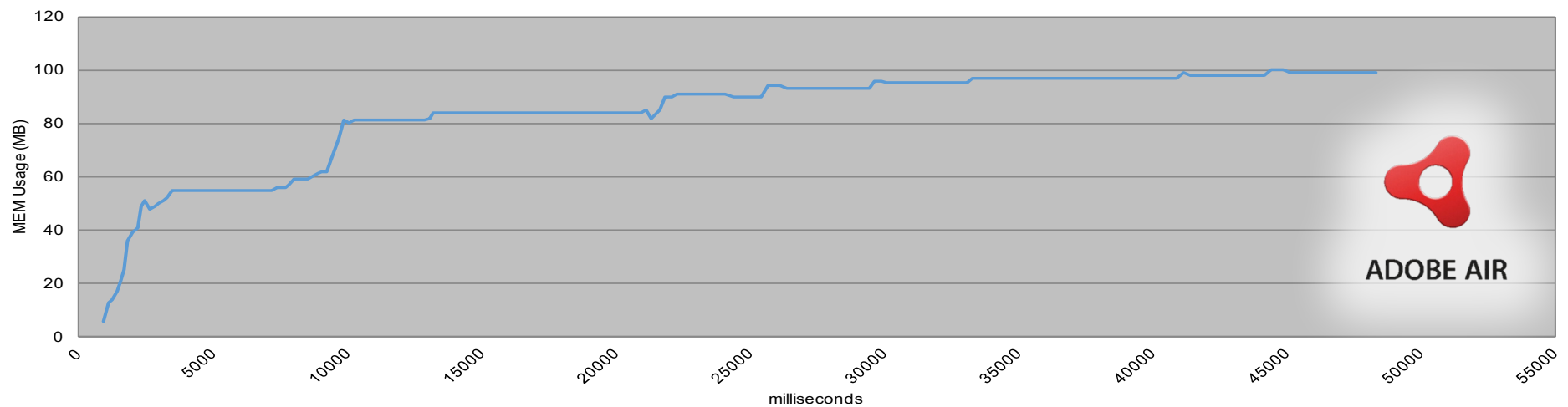
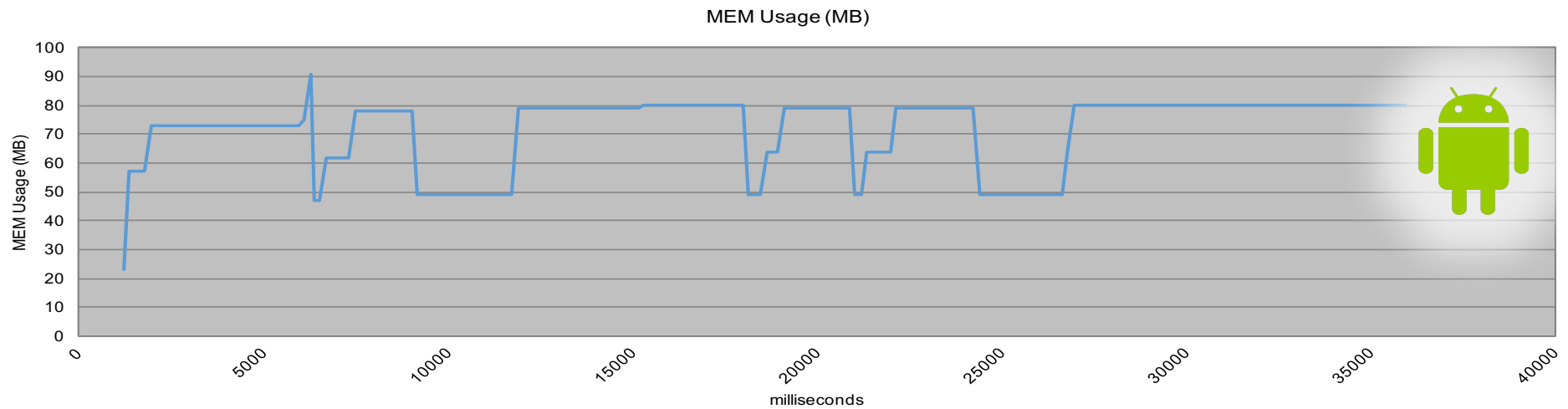
CPU Usage



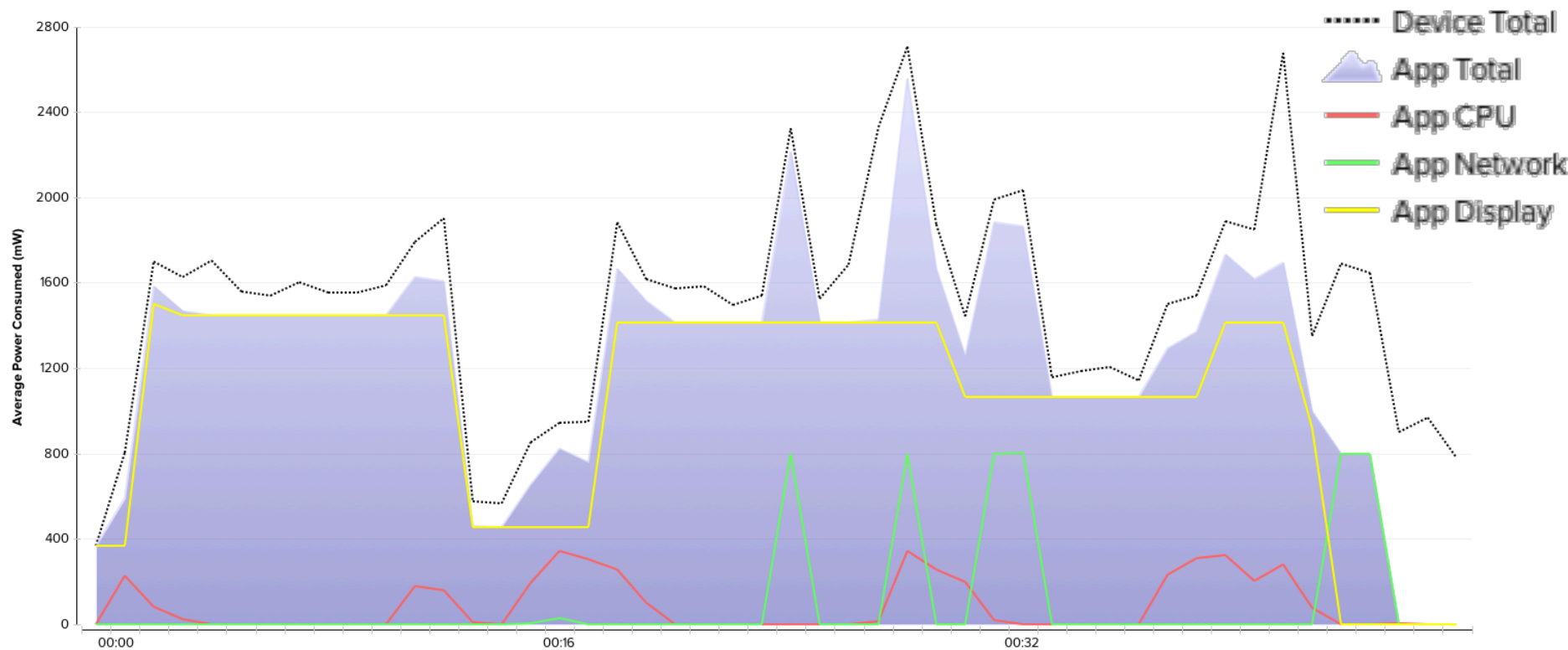
Memory Usage



Memory Usage



Battery Usage





Q S A