Sistemi Embededd
a.a. 2013/2014

La piattaforma Android

Raffaele Di Natale
raffaele.dinatale@dieei.unict.it

Bibliografia

Web sites

- http://developer.android.com/

E-books

- http://it-ebooks.info/book/659/
- http://www.sprik.it/guida/Android4_2.pdf (in italiano)

Books

- Android Guida per lo Sviluppatore, APOGEO
- Beginning Android 4, APRESS

Agenda

- **Introduzione ad Android**: contesto, storia e confronto con altre piattaforme, caratteristiche
- **Architettura**: Applicazioni, Application Framework, Librerie, Android Runtime, ...
- **Dalvik Virtual Machine**: confronto con JVM, gestione della memoria, bytecode
- **Strumenti di Sviluppo**: SDK, Emulator
- **Analisi di una App:** Building blocks e ciclo di vita
- **Esempi**: utilizzo componenti, sviluppo e debug di applicazioni

Introduzione ad Android

La rivoluzione dei dispositivi mobili 1/2

Il dispositivo che fino a qualche anno fa chiamavamo cellulare oggi è a tutti gli effetti un Personal Computer.

La vera rivoluzione dal punto di vista degli utenti consiste nel fatto che uno smartphone è molto più Personal di un PC perché ci segue ovunque.

Gli sviluppatori possono finalmente creare applicazioni che non devono fare più i conti con il concetto di "risorse limitate".

La rivoluzione dei dispositivi mobili 2/2

Dal punto di vista dei produttori la rivoluzione consiste nel fatto che in passato ogni produttore di dispositivi mobili ha realizzato un proprio sistema operativo, ambiente di sviluppo, linguaggio, tool: nessuno si è affermato come standard.

Il progetto Android nasce quindi da una esigenza di standardizzazione su iniziativa di Google.

http://www.android.com/

Cosa è Android?

Android è la prima piattaforma software open per dispositivi mobile

Android è uno stack software per dispositivi mobili ed include un sistema operativo, un middleware ed un insieme di applicazioni chiave.

Android è stato sviluppato da Google Inc. e da un consorzio di aziende chiamato Open Handset Alliance

Perchè Open?

E' open perchè:

- Utilizza tecnologie open (linux kernel 2.6 e 3.x dalla 4.0)
- Le librerie e le API utilizzate per realizzare Android sono le stesse che possiamo usare per le nostre applicazioni

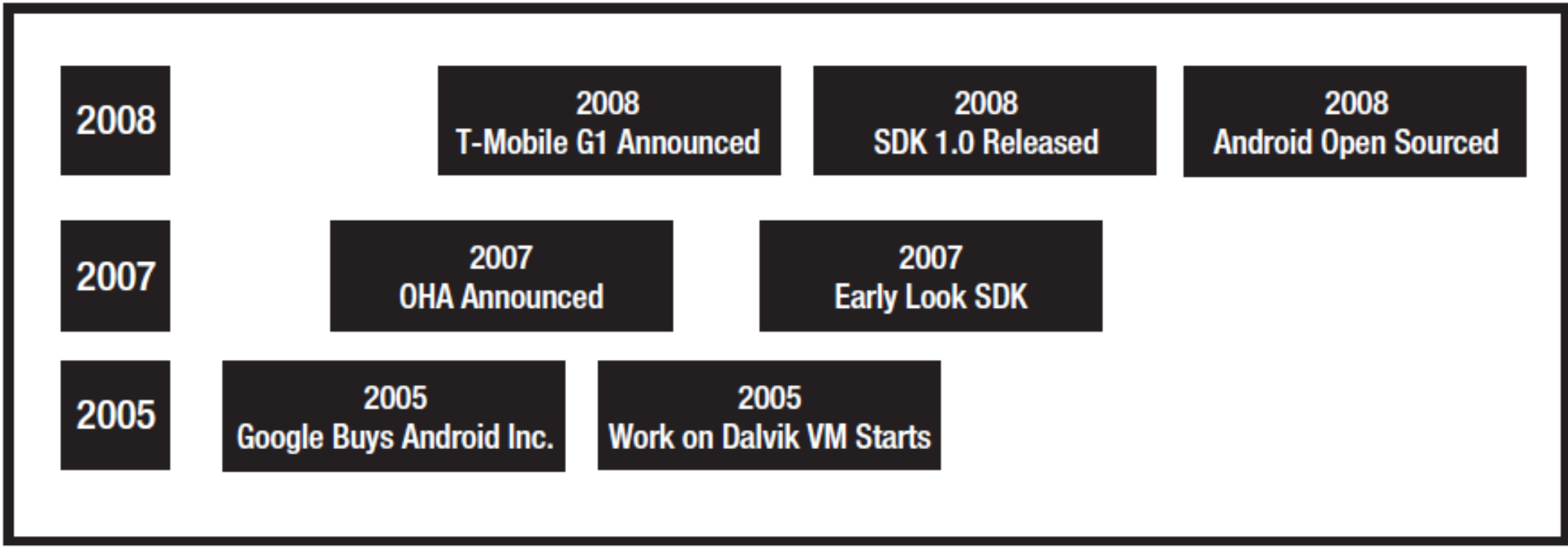
Il kernel linux è rilasciato sotto GNU General

Open Handset Alliance

http://www.openhandsetalliance.com

- Operatori mobili: Telecom, Vodafone, T-Mobile...
- Produttori di chip: Arm, Intel, NVIDIA...
- Produttori di telefonini: Toshiba, HTC, Acer, Sony,...
- Aziende sviluppatrici di software: Google, eBay, Omron, ...
- Aziende di commercializzazione: Accenture, Aplix...

Un po' di storia



G1 T-Mobile

Processor Qualcomm® MSM7201A™, 528 MHz

RAM: 192 MB

Storage: Flash Memory 256 MB

Display 3.2"

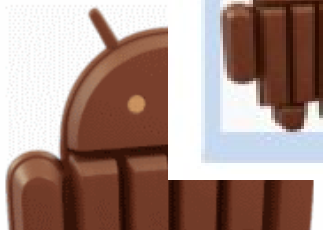
Supporto 3G UMTS/HSDPA a 7,2 Mbps

Platforms 1/2

	Platform	Codename	API Level
	1.0	BASE	1
	1.1	BASE_1_1	2
	1.5	CUPCAKE	3
	1.6	DONUT	4
	2.1	ECLAIR_0_1	7
	2.2	FROYO	8
	2.3 - 2.3.2 2.3.3 - 2.3.7	GINGERBREAD	9 10

Platforms 2/2

	Platform	VERSION_CODE	API Level
	3.1	HONEYCOMB_MR1	12
	3.2	HONEYCOMB_MR2	13
	4.0.3 - 4.0.4	ICE_CREAM_SANDWICH _MR1	15
	4.1, 4.1.1	JELLY_BEAN	16
	4.2, 4.2.2	JELLY_BEAN_MR1	17
	4.3	JELLY_BEAN_MR2	18
	4.4	KITKAT	19



Requisiti Minimi 1/2

I requisiti minimi di ogni versione rilasciata, sono definiti mediante l'Android Compatibility Program che consiste di tre componenti chiave:

- Il codice sorgente dello stack software Android
- Il “Compatilbility Definition Document”, che descrive le specifiche di compatibilità
- La “Compatilbility Test Suite”, che rappresenta il "meccanismo" di compatibilità
- Il più recente ACP si riferisce alla 4.3

Requisiti Minimi 2/2

Per Android 2.2 alcuni dei requisiti minimi sono:

Memoria >= 128MB (kernel) + 150MB (/data) +
1GB (Application/SD Card)
FotoCamera >= 2 megapixel
Display >= 2,5"

Per Android 4.0 alcuni dei requisiti minimi sono:

Memoria >= 340MB (kernel) + 350MB (/data) +
1GB (Application(SD Card)
FotoCamera >= 2 megapixel
Display >= 2,5"

Codice Sorgente di Android

- Il codice sorgente di Android può essere scaricato seguendo le indicazioni descritte in
  - http://source.android.com/source/downloading.html
- La versione di Android più recente per la quale è disponibile il codice sorgente è la 4.3 (API level 18)

Caratteristiche principali 1/3

- Android include un ”application framework” che permette il riutilizzo dei componenti sviluppati.
- La Dalvik Virtual Machine è ottimizzata per l'esecuzione su dispositivi mobili.
- Il browser web integrato è basato sul software open-source ”WebKit”.
- La grafica 2D è basata su una libreria appositamente sviluppata.

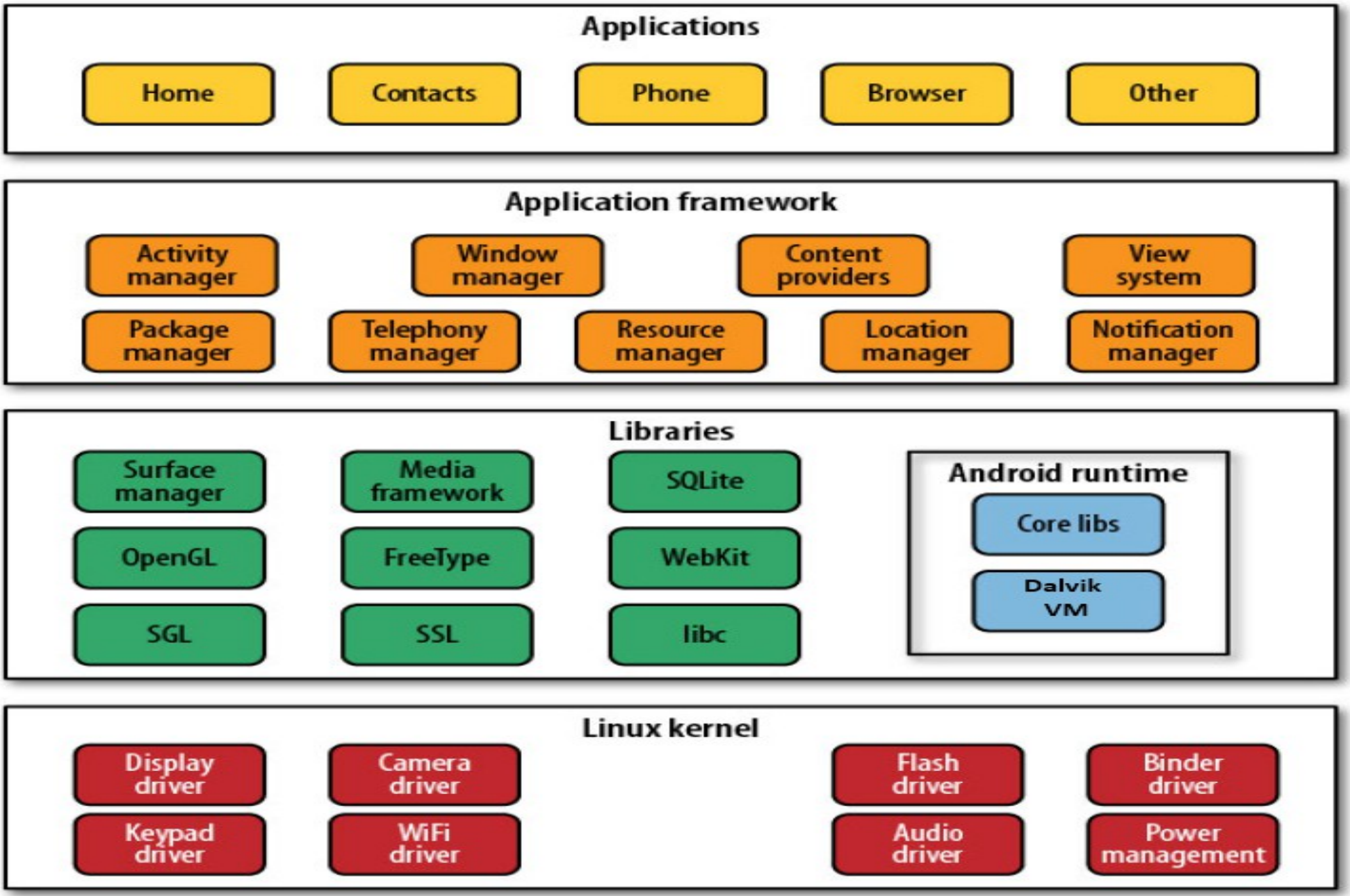
Caratteristiche principali (2/3)

- La grafica 3D è basata sulla specifica OpenGL ES 1.0 e l'accelerazione hardware è opzionale.
- Per l'immagazzinamento strutturato di dati si usa SQLite.
- Sono supportati i più comuni formati audio e video (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- Supporta la telefonia GSM.

Caratteristiche principali (3/3)

- A seconda dell'hardware può supportare anche EDGE (2.5G), UMTS (3G), LTE(4G), Bluetooth/BTLE (Low Energy) e Wi-Fi.
- Prevede anche il supporto per fotocamera, GPS, bussola e accelerometro.
- Fornisce un ricco ambiente integrato di sviluppo che include un emulatore, strumenti per il debugging e il profiling di memoria e prestazioni, ed un plugin per l'IDE Eclipse per sviluppare applicazioni in linguaggio Java.

Architettura di Android



Componenti del sistema

I componenti mostrati nel precedente schema dell'architettura verranno ora analizzati uno ad uno:

- Applicazioni
- Application Framework
- Librerie
- Android Runtime
- Kernel Linux

Applicazioni

Android dispone di base di un insieme di applicazioni che includono un client email, un programma per la gestione degli SMS, agenda, rubrica, mappe, un browser web e altre.

Tutte le applicazioni vengono scritte utilizzando il linguaggio di programmazione Java.

Application Framework

- Gli sviluppatori hanno pieno accesso alle stesse API del framework che vengono utilizzate dalle applicazioni di base.
- L'architettura delle applicazioni è progettata per semplificare il riutilizzo dei componenti.
- Ogni applicazione può pubblicare le sue capabilities e le altre possono farne uso.
- Questo stesso meccanismo fa sì che l'utente possa rimpiazzare alcuni componenti.

Servizi dell'Application Framework (1/2)

L'Application Framework fornisce alle applicazioni i seguenti servizi:

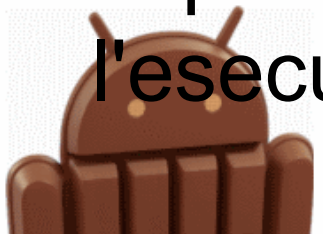
- Un insieme di **Views** completo ed estensibile che include liste, griglie, text-box, pulsanti e perfino un browser web integrato nell'applicazione.
- I **Content Provider** permettono alle applicazioni di accedere ai dati di altre applicazioni (es. la rubrica) o di mettere in condivisione i loro stessi dati.

Servizi dell'Application Framework (2/2)

- Un **Resource Manager** che fornisce l'accesso alle risorse esterne al codice, come stringhe localizzate, grafici e file di layout.
- Un **Notification Manager** permette a tutte le applicazioni di mostrare avvisi personalizzati sulla barra di stato.
- Un **Activity Manager** che gestisce il ciclo di vita delle applicazioni.

Librerie Native (1/4)

- Sono librerie scritte in C/C++ che rappresentano il core di Android e sono usate dai vari componenti del sistema.
- Le capabilities di queste librerie sono esposte agli sviluppatori sempre attraverso l'Application Framework.
- La libreria più importante è senz'altro la **Standard C Library** (libc) derivata dalla sua implementazione BSD e ottimizzata per l'esecuzione su dispositivi embedded.



Librerie Native (2/4)

- Il **Media Framework** (librerie multimediali) sono basate su OpenCORE di PacketVideo e supportano la riproduzione e la registrazione di vari formati audio/video.
- Il **Surface Manager** gestisce l'accesso al sottosistema di visualizzazione su schermo e senza sforzo da parte del programmatore compone livelli di grafica 2D e 3D provenienti da varie applicazioni.

Librerie Native (3/4)

- **WebKit** (LibWebCore) è il motore di browser alla base sia del browser principale di Android che delle web view integrate nelle applicazioni.
- **SGL** è l'engine grafico 2D.
- Le **librerie 3D** implementano la API OpenGL ES 1.0 e 2.0 e possono utilizzare sia l'accelerazione hardware (se disponibile) sia una implementazione software ottimizzata.

Librerie Native (4/4)

- La liberia **FreeType** fornisce il rendering per i font bitmap e vettoriali
- **SQLite** è un potente e leggero motore di database relazionale disponibile per tutte le applicazioni, per la memorizzazione di qualsiasi tipo di dato strutturato
- **SSL** è la libreria per la gestione dei Secure Socket Layer.

Android Runtime

- L'Android Runtime si compone delle Core Libraries e della Dalvik Virtual Machine.
- Le **Core Libraries** forniscono molte delle funzionalità delle analoghe librerie disponibili per il linguaggio di programmazione Java.
- Ogni applicazione Android gira in un suo proprio processo, con la sua istanza della **Dalvik Virtual Machine** (VM).

Core Libraries

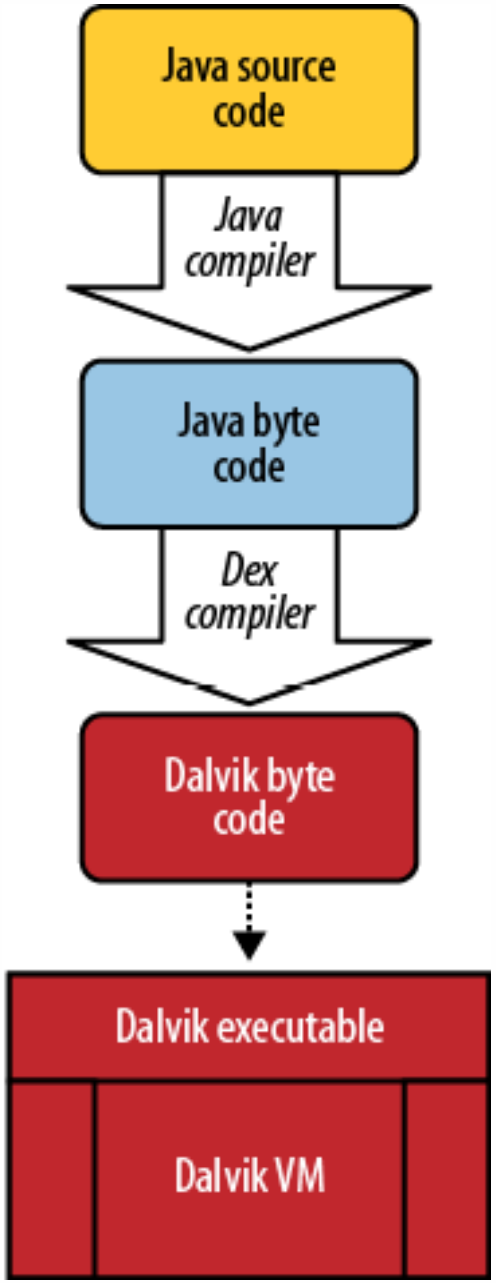
- Rappresentano un insieme di librerie molto ”vicine” alla Dalvik Virtual Machine ed estratte in parte dal progetto Apache Harmony (2007) ed in parte appositamente adattate
- Sono distribuite in tre categorie:
    - DalvikVM specific libraries (System Info, Debugging)
    - Java Compatibility libraries (Base and utility classes)
    - Thirty Part Libraries (Apache HttpClient 4.0)

Dalvik VM (1/2)

- La Dalvik VM è stata ottimizzata in modo che uno stesso dispositivo embedded sia in grado di eseguirne più istanze in modo efficiente.
- La Dalvik VM esegue un bytecode incompatibile con quello della Java VM e che viene detto Dalvik Executable (.dex), ottimizzato per avere una occupazione di memoria minima.
- La Dalvik VM è register-based (mentre la Java VM è stack-based).

Dalvik VM (2/2)

Il ciclo di sviluppo prevede che una applicazione venga dapprima sviluppata in Java, e successivamente un tool (dx) converte i file .class, ottenuti dalla normale compilazione java, in file ottimizzati .dex



Kernel Linux

- Dalla versione 4.0 Android si basa sul kernel Linux 3.x per i servizi di base come la sicurezza, la gestione della memoria, la gestione dei processi, lo stack di rete e il modello dei driver.
- Il kernel funge anche da livello di astrazione tra l'hardware e il resto dello stack software. In questo livello avviene la gestione dei driver dell'hardware (videocamera, display, memoria flash, la scheda wireless...)

Piattaforme per Dispositivi Mobili

Piattaforme per dispositivi mobili (Passato)

- 1996
  - Palm OS
  - Windows CE
- 2000
  - Symbian
  - Windows Pocket PC
- 2006
  - Java ME

Piattaforme per dispositivi mobili (Presente)

- Windows Phone 8
- iOS 7

- Blackberry (Java ME + API BlackBerry)
- Android

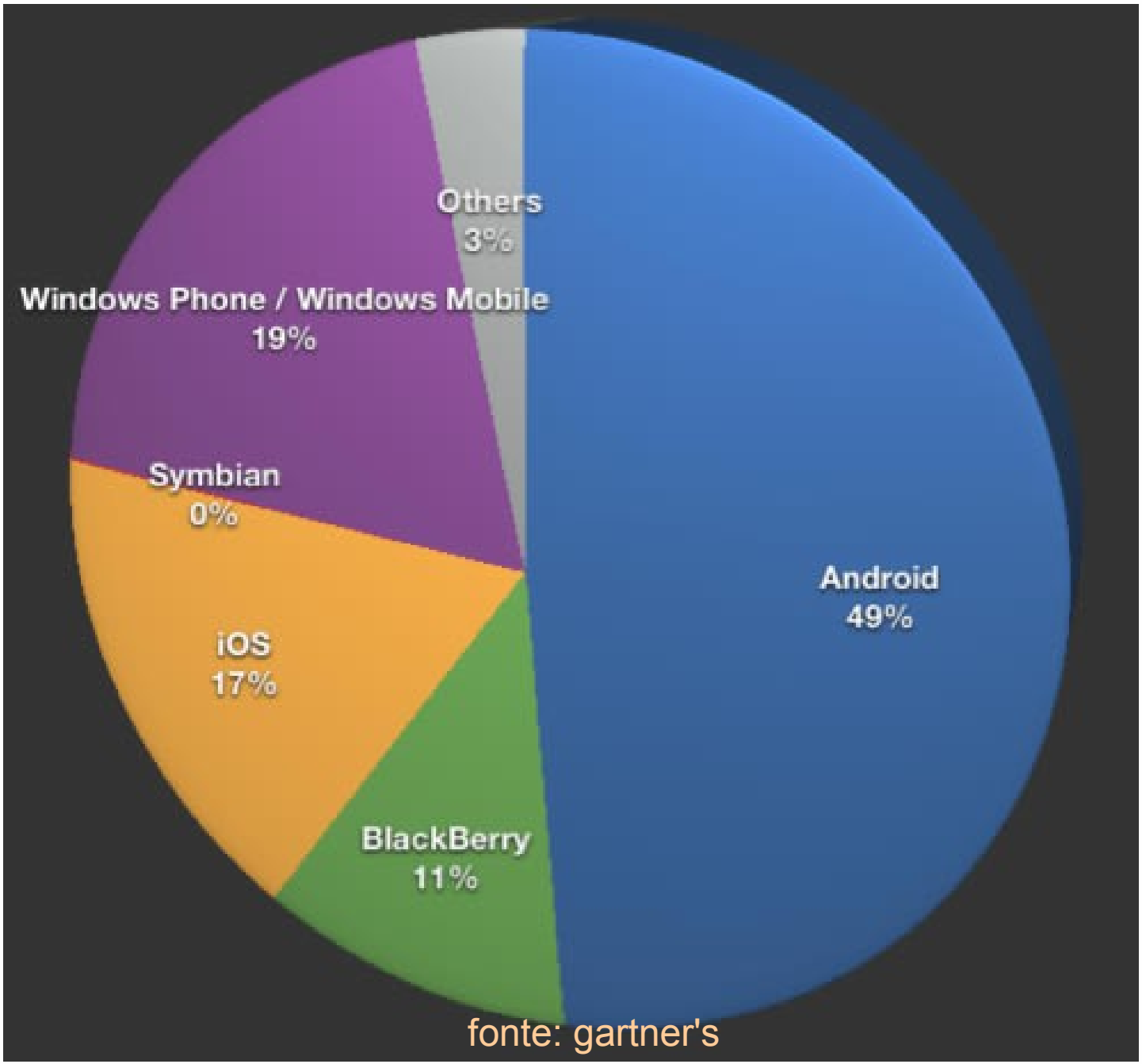
Mercato Mondiale 2012-2013

Top Four Operating Systems, Shipments, and Market Share, Q3 2013 (Units in Millions)

Operating System	3Q13 Shipment Volumes	3Q13 Market Share	3Q12 Shipment Volumes	3Q12 Market Share	Year-Over-Year Change
Android	211.6	81.0%	139.9	74.9%	51.3%
iOS	33.8	12.9%	26.9	14.4%	25.6%
Windows Phone	9.5	3.6%	3.7	2.0%	156.0%
BlackBerry	4.5	1.7%	7.7	4.1%	-41.6%
Others	1.7	0.6%	8.4	4.5%	-80.1%
Total	**261.1**	**100.0%**	**186.7**	**100.0%**	**39.9%**

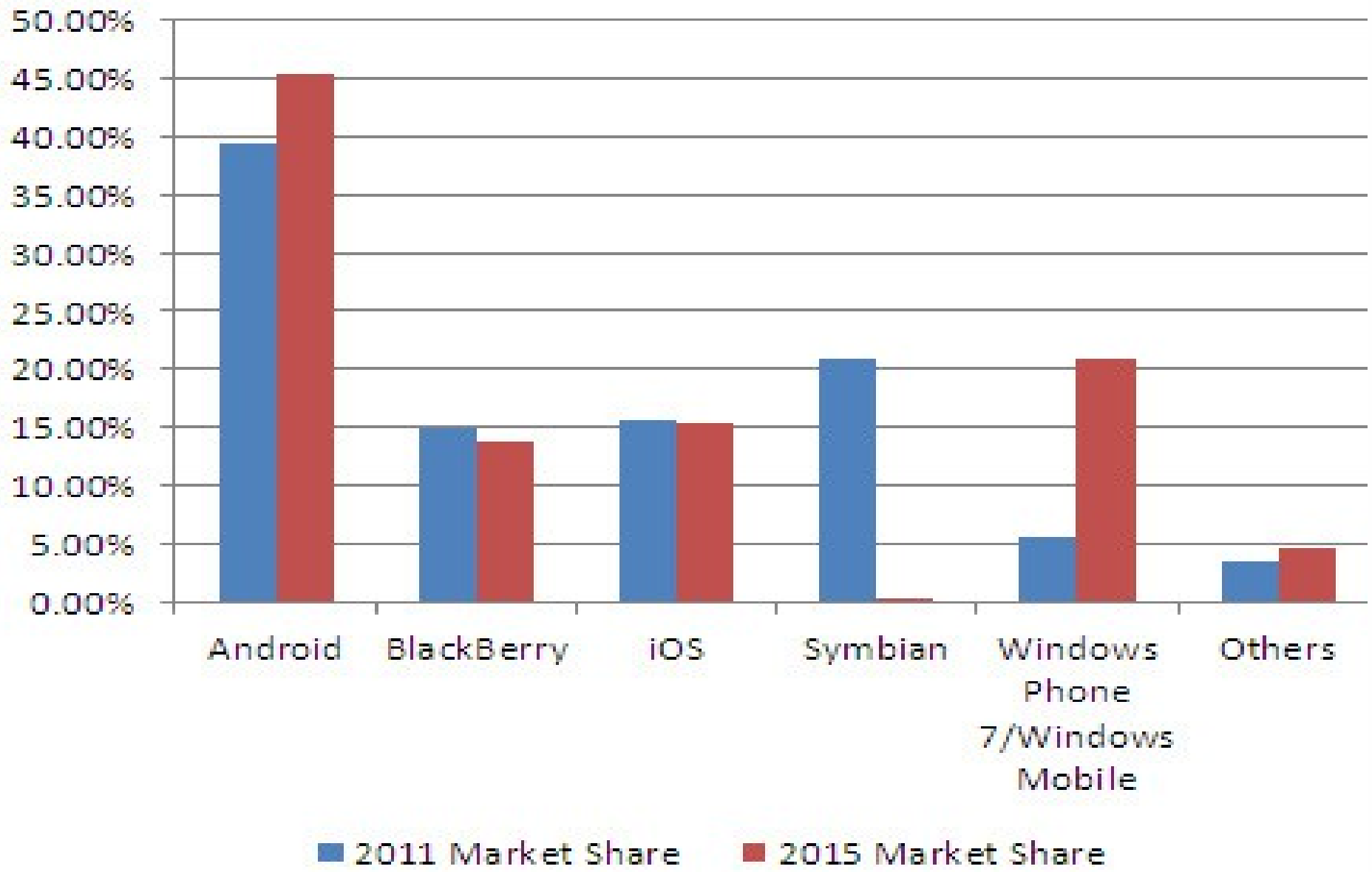
fonte: IDC

Previsione Mercato USA 2015



fonte: gartner's

Previsione mondiale 2011-2015



fonte: idc.com

Blackberry

- I telefoni Blackberry di RIM (Research-In-Motion) sono prodotti diffusi da anni nell'ambito enterprise.
- Sono specializzati nella gestione della posta elettronica in modalità ”push”.
- Sebbene siano stati considerati prodotti all'avanguardia, sono sempre stati utilizzati solo nell'ambito aziendale e non hanno quindi avuto una grande diffusione nel mercato consumer che garantisce i volumi più elevati.
- Nel 2013 ha dovuto affrontare una grave crisi ed è stata ceduta ad una nuova azienda

Sistemi Operativi per Smartphone

- Di conseguenza nei prossimi anni la vera lotta tra i sistemi operativi per smartphone avverrà fra tre grandi concorrenti: Windows Phone, iOS e Android.
- Dietro questi sistemi operativi si celano i colossi del mondo IT, ovvero Microsoft (Nokia), Apple, Google, Samsung.

Confronto tra SO:strumenti di sviluppo

- Per tutte le piattaforme prese in considerazione l'SDK che permette lo sviluppo di applicazioni è liberamenente disponibile.
- Riguardo ai linguaggi di programmazione da adoperare, per Windows Mobile è possibile utilizzare indifferentemente C++, C# e VB.NET.
- Per iOS si usa Objective-C.
- Per Android e BlackBerry si usa Java.

Confronto tra SO: distribuzione applicazioni

- Tutte le piattaforme provedono: un App Store ufficiale (Apple-iTunes App Store, Android Market, Nokia-Ovi Store, Windows Phone MarketPlace e BlackBerry-App World) dal quale è possibile scaricare programmi gratuiti o a pagamento.
- L'installazione tramite PC è prevista da Windows mediante il WMDC, Nokia con il Nokia OVI Suite, Android Platform per Android e BlackBerry mediante il Desktop Manager .

Piattaforme Mobile in arrivo

- Open Source – Linux based

- Open Source – Linux based

- Closed Source, Cloud based – Android Fork

Android VS Java

Linguaggio Java e VM

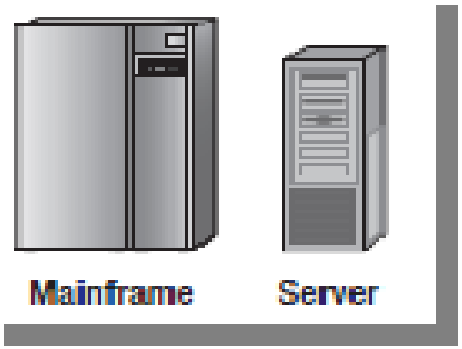
- Quando si parla di Java bisogna subito chiarire se ci si riferisce al linguaggio di programmazione oppure alle varie versioni di librerie e Virtual Machine.
- Dal punto di vista del linguaggio di programmazione, Android si basa al 100% sul linguaggio di programmazione Java ed in particolare richiede la presenza del JDK 6 per permettere lo sviluppo con Eclipse o altri IDE.

Non solo Java

Google fornisce agli sviluppatori due ulteriori strumenti per lo sviluppo di applicazioni:

- Scripting Layer for Android (SL4A): supporto ai più diffusi linguaggi di script (JavaScript, Python, Perl, JRuby, Lua) per lo sviluppo di applicazioni
- Android Native Development Kit (NDK): sviluppo di applicazioni in codice nativo (C/C++)

Versioni di Java 1/2



JAVA EE
Enterprise
Edition

per la creazione di servlet per il mondo dei web service

JAVA SE
Standard
Edition

per la creazione di applet e applicazioni per il mondo desktop

JAVA ME
Connected
(CDC)

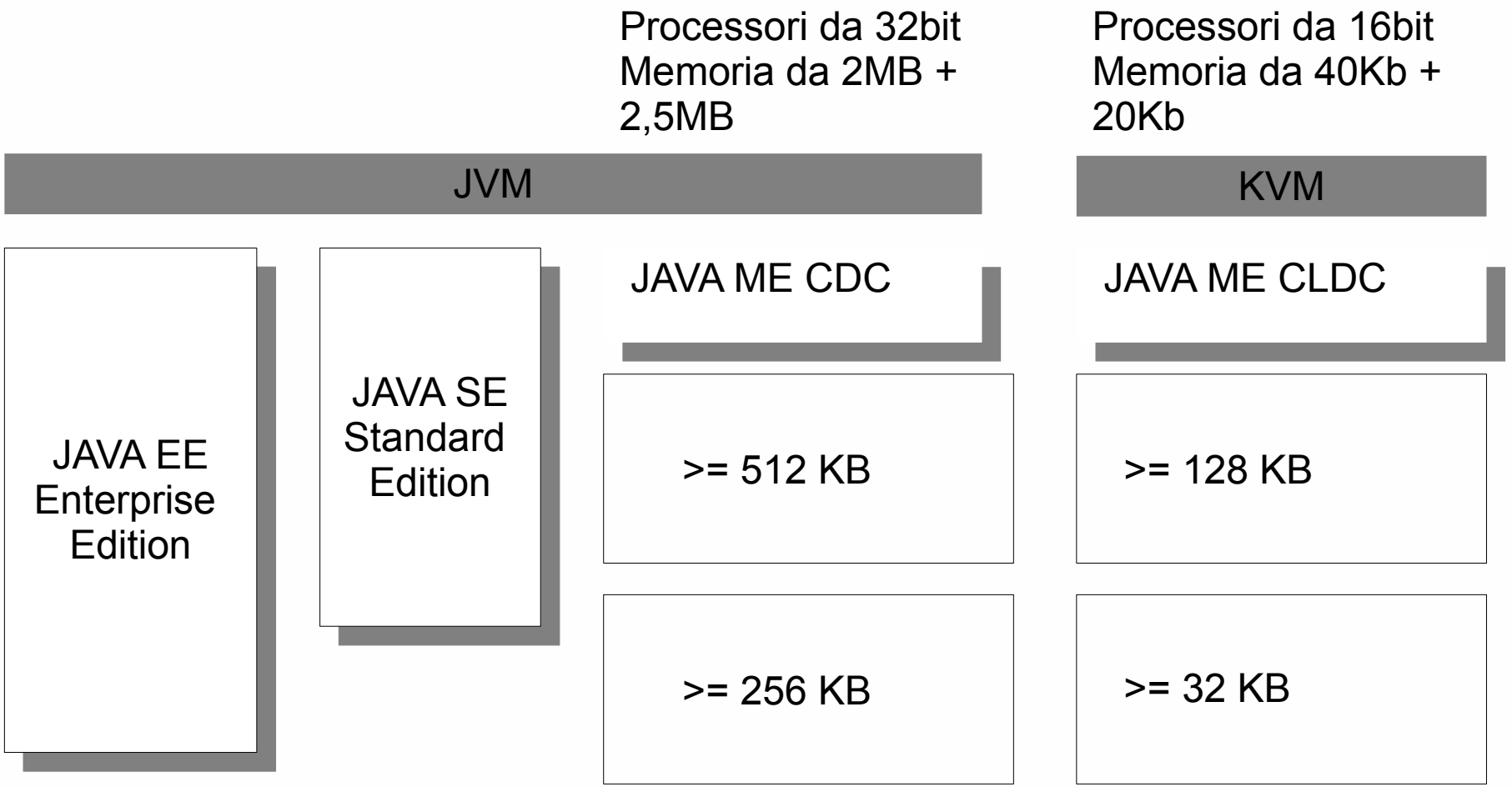
per la creazione di Xlet per dipositivi mobile sempre connessi

JAVA ME
Limited
(CLDC)

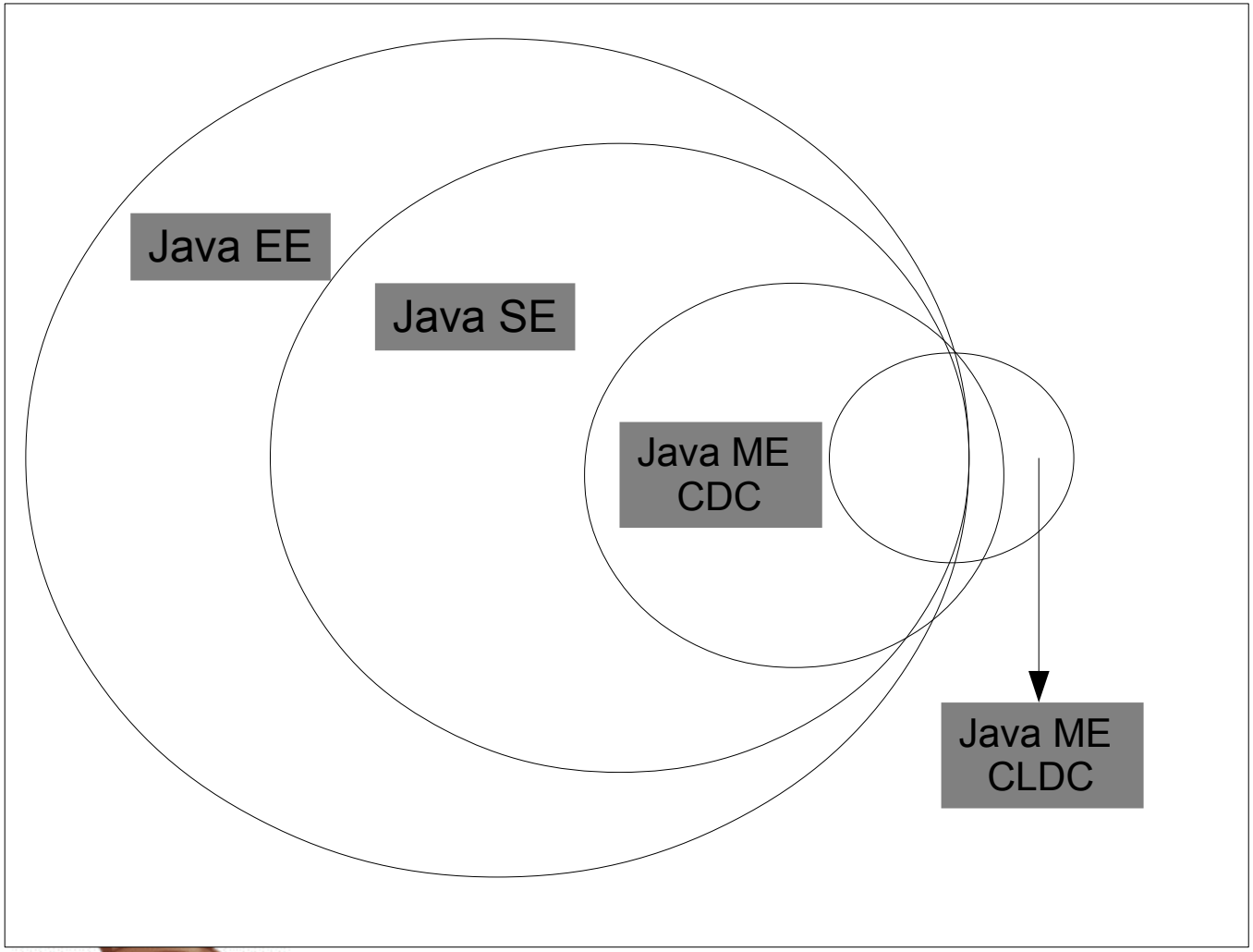
per la creazione di Midlet per dispositivi mobile limitatamente connessi

CDC = Connected Device Configuration
CLDC = Connected Limited Device Configuration

Versioni di Java 2/2



API Java



CDC e CLDC possono supportare API Java esterne a Java SE. Come conseguenza si potrebbe avere un programma Java che viene eseguito correttamente su Desktop, ma non vi sono garanzie che sia eseguito anche su dispositivi che supportano solo ME

JavaME e Android 1/3

- Configurazioni: JavaME supporta due classi di micro-dispositivi e offre soluzioni standardizzate per ognuna di esse . Android supporta una sola classe e non è possibile eseguirlo su dispositivi ”limitati”
- Comprensibilità: Android è pensato per un unico modello di dispositivo e quindi utilizza una UI ”adattabile”. JavaMe dispone di modelli di UI differenti per device: MIDlets, Xlets, AWT e Swing: include più di 20 JSR (Java Specification Request)

JavaME e Android 2/3

- Reattività: La DVM è maggiormente ottimizzata e più reattiva se paragonata alla JVM su uno stesso dispositivo. In realtà dovremmo paragonare la DVM alla KVM ma ciò non è possibile perchè questa è pensata per dispositivi di basso profilo.
- Compatibilità con JAVA: a causa della DVM, Android esegue solo bytecode .dex e non java: è sempre possibile compilare il codice Java in standard Java Class, pur non essendo possibile eseguirne direttamente il bytecode.

JavaME e Android 3/3

- Scelta: JavaME è ampliamente supportato in quanto diffuso su moltissimi dispositivi mobili, ma l'uniformità, il costo e la facilità nello sviluppo fanno migrare ogni giorno molti programmatori Java verso Android.
- Supporto JavaSE: se paragonato al supporto di JAVAME-CDC, il supporto di Android a JavaSE è più completo, fatta eccezione per AWT e Swing.

Android e Dalvik VM

- Google ha quindi preso un sottoinsieme delle librerie JAVA, (poi delle Apache Harmony - 2007), ha aggiunto alcuni propri package specifici per l'utilizzo in dispositivi mobili e ha definito un proprio formato bytecode ottimizzato per ridurre al minimo l'occupazione di memoria.
- La macchina virtuale in grado di eseguire questo bytecode come detto si chiama Dalvik VM, e non essendo una macchina virtuale Java ME non è previsto il pagamento di nessuna royalty.

Confronto tra le API

- Cercheremo adesso di fare un rapido confronto tra le API di Java SE e quelle di Android.
- Elencheremo prima quelle supportate anche in Android che possono essere quindi utilizzate senza modificare eventuale codice già scritto in linguaggio Java.
- Successivamente elencheremo quelle che sono state rimosse e magari sostituite da altre ritenute migliori.

API estese di Java SE supportate in Android

- `javax.crypto`
- `javax.net`
- `javax.security`
- `javax.sound`
- `javax.sql`
- `javax.xml.parsers`
- `org.w3c.dom (tranne i sotto-package)`
- `org.xml.sax`

API di Java SE NON supportate in Android 1/2

- `java.applet`
- `java.awt`
- `java.beans`
- `java.lang.management`
- `java.rmi`

API estese di Java SE NON supportate in Android (1/2)

- `javax.accessibility`
- `javax.activity`
- `javax.imageio`
- `javax.management`
- `javax.naming`
- `javax.print`
- `javax.rmi`
- `javax.security.auth.kerberos`
- `javax.security.auth.spi`
- `javax.security.sasl`

API estese di Java SE NON supportate in Android (2/2)

- `javax.swing`
- `javax.transaction`
- `javax.xml (tranne javax.xml.parsers)`
- `org.ietf.*`
- `org.omg.*`
- `org.w3c.dom.* (i sotto-package)`

Java ME

- Java ME viene attualmente considerato quasi obsoleto, in quanto presenta potenzialità molto ridotte e non supporta molte delle tecnologie che sono state sviluppate negli ultimi anni.
- Sebbene sia stato sviluppato da Sun Microsystems con molte ambizioni, non è mai riuscito ad uscire dalla nicchia dei videogiochi.
- Non viene considerato un sistema operativo per ”smartphone”, ovvero telefoni con funzionalità simili a quelle di un PC.

La macchina virtuale Dalvik

Perché una VM?

- Quando nacque Java venne coniato lo slogan ”Write once, run everywhere”, ovvero ”scrivi il programma una volta e poi lo esegui su qualunque dispositivo”.
- Questo avviene perché la macchina virtuale è uno strato software che astrae l'hardware sottostante, mettendo a disposizione del bytecode una macchina sempre uguale.
- Android, come anticipato usa un approccio simile, e la sua VM si chiama Dalvik.

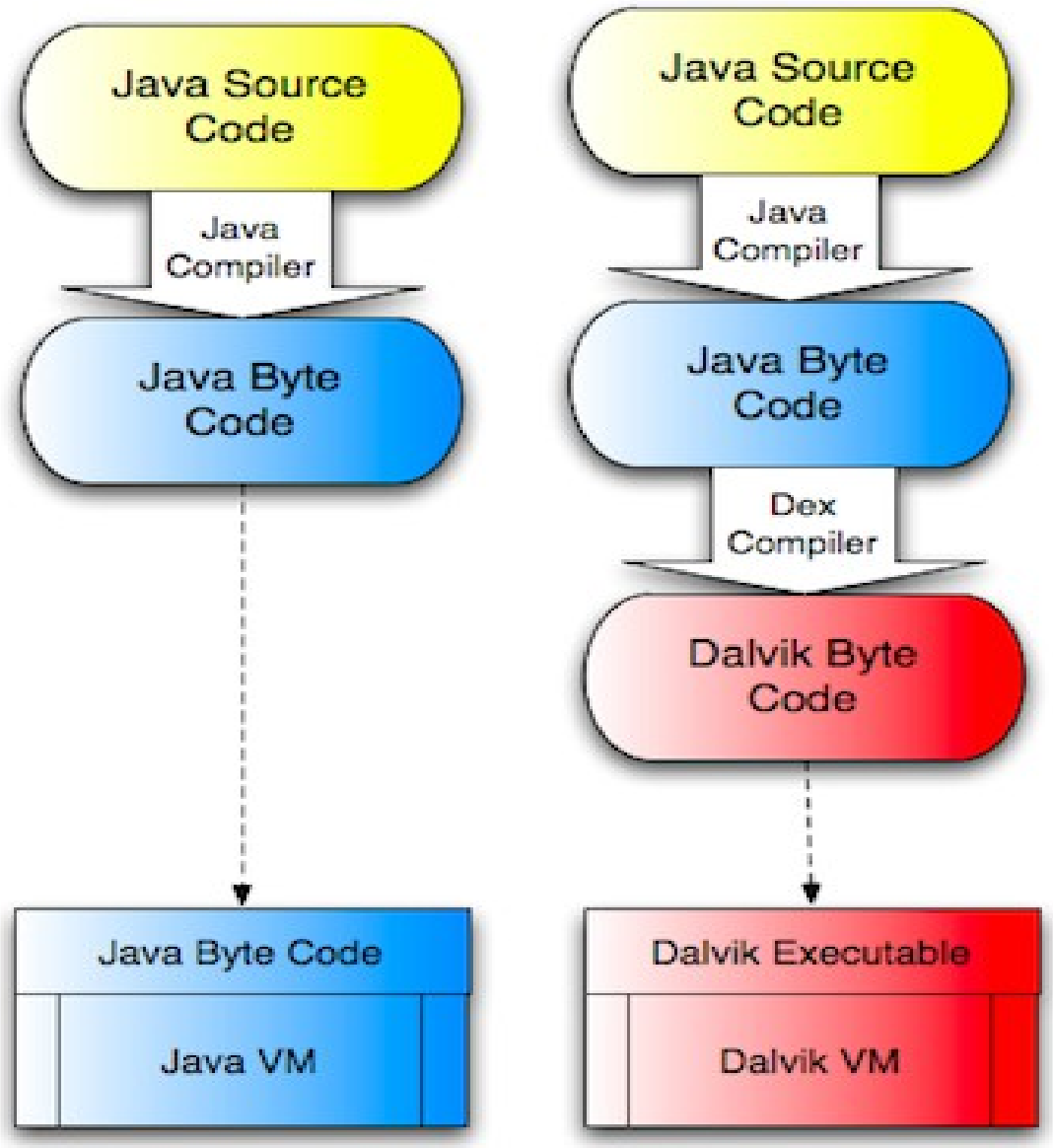
Requisiti per Dalvik

- La Dalvik VM è stata progettata da Dan Bornstein, ingegnere di Google originario della città di Dalvik in Islanda.
- I requisiti che a Dan Bornstein sono stati imposti per la progettazione di questa Virtual Machine sono stati la capacità di girare su un sistema con una CPU lenta, con poca memoria, con un sistema operativo senza area di swap e il tutto alimentato solo da una batteria ed un application runtime di tipo “sandbox”.

Il problema della memoria

- Inizialmente, nella progettazione della DVM, si è supposto di avere a disposizione un tipico smartphone con soli 64 MByte di memoria.
- Questo spazio dopo l'avvio del kernel Linux si riduce a 40 MByte, che diventano 20 dopo l'avvio dei servizi di alto livello.
- Si è visto inoltre che la libreria C standard (libc) è molto vasta e occupa circa 10 Mbyte.

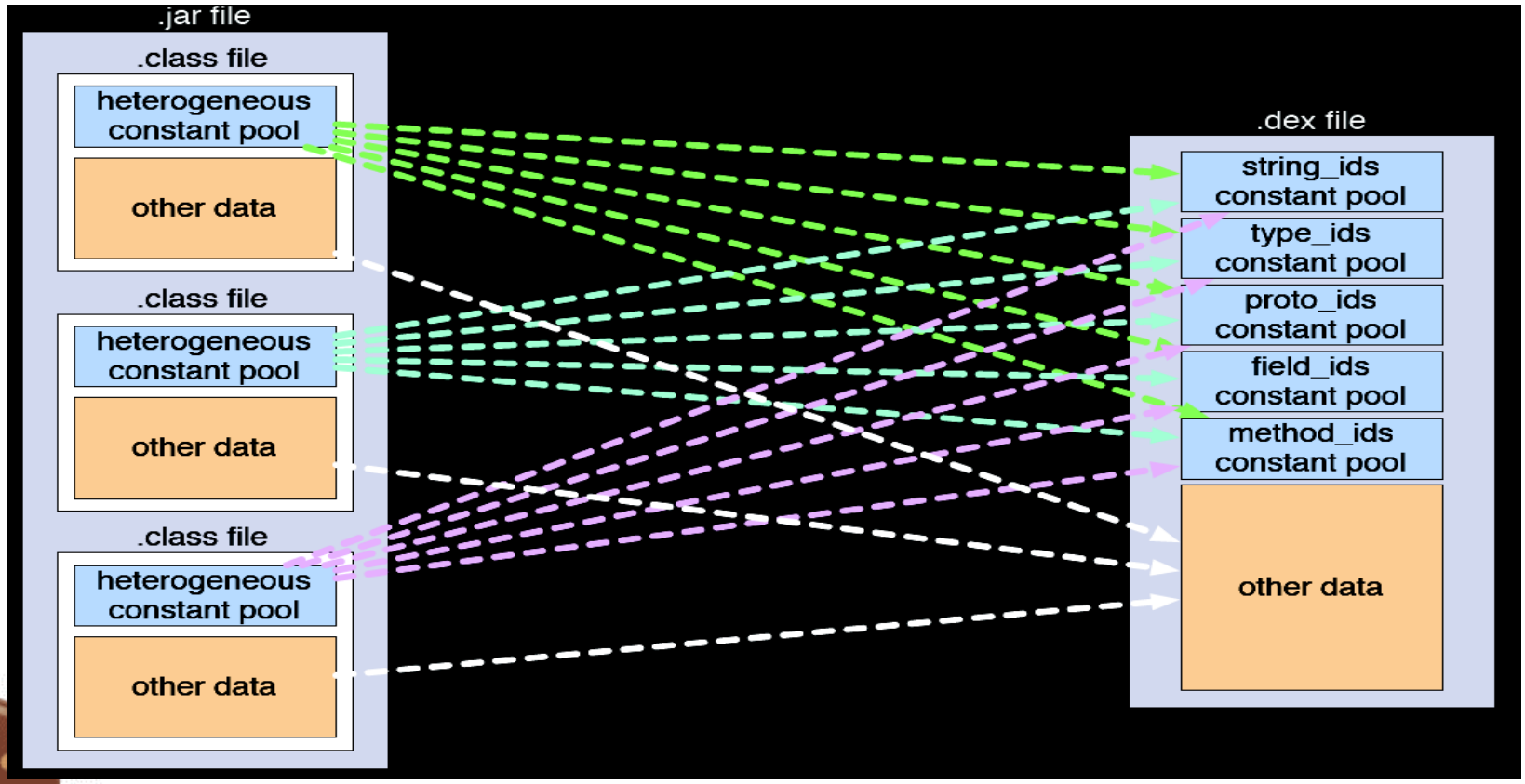
Java versus Dalvik



fonte: oreilly.com

Dalvik Executable

- Un primo lavoro che è stato fatto è stato quello di specificare un formato (dex o Dalvik Executable) in grado di risparmiare spazio rispetto al bytecode di Java:



Confronto tra bytecode

- Utilizzando questo nuovo formato si ottengono già dimensioni del bytecode molto ridotte.
- Ad esempio le librerie di sistema, rispetto al 100% originario, diventano grandi il 50% quando compresse da Java in formato JAR e 48% in formato dex non compresso (non richiede decompressione).
- L'applicazione browser passa invece dal 100% originario e 49% compresso in JAR a solo il 44% in formato dex non compresso.

Separazione della memoria

- Un altro accorgimento adottato in Dalvik rispetto a Java è la separazione della memoria nel momento in cui avviene una chiamata di sistema fork().
- A differenza di quanto accade in Java, ogni applicazione Android ha un proprio processo ed una propria istanza della DVM.
- La DVM è stato progettata in modo tale che un dispositivo possa eseguire molte VM in maniera efficiente.
- Anche il meccanismo di Garbage Collector di ogni applicazione deve essere indipendente.

Efficienza

- Per quanto riguarda la CPU si è supposta una frequenza compresa fra i 250 e i 500 MHz, con un bus di sistema a 100 MHz e una cache dati di 16 o 32 KByte.
- Inizialmente (fino alla 2.1) la DVM non supportava la compilazione Just-In-Time (JIT) perché si pensava non necessaria poiché per i calcoli più impegnativi (grafica e codec audio/video) erano presenti dei coprocessori ed in ultima istanza si poteva ricorrere alla JNI (Java Native Interface).
- Dalla versione 2.2 di Android la DVM possiede un Just-In-Time (JIT) compiler.

Installazione

Nel momento in cui si installa nel sistema una nuova applicazione Android vengono eseguite le seguenti operazioni:

- Per sicurezza si effettua la verifica che tutti gli indici interni ai file dex siano corretti.
- Anche se non si utilizzano tecniche JIT il codice viene ottimizzato ad esempio spostando i dati dove conviene averli e ricopiando alcune funzioni che conviene avere ”inline”.

VM basata su stack o su registri?

- La Java VM è stack-based, ovvero tutte le operazioni agiscono sugli operandi che stanno sulla cima dello stack.
- La Dalvik VM è invece stata progettata register-based, ovvero le operazioni agiscono su alcuni registri virtuali.
- Le statistiche di Google dicono che in questo modo si esegue il 30% di istruzioni in meno.
- Anche il numero di accessi in memoria si riduce.

Sicurezza

- Un'applicazione, per default, non ha il permesso di eseguire alcuna operazione che può avere impatto negativo su altre su altre applicazioni, sul sistema operativo o sull'utente.
- Ciò comprende la lettura e la scrittura di dati privati dell'utente (contatti, e-mail, ...), leggere o scrivere i file relativi al altre applicazioni, accedere alla rete, mantenere il dispositivo attivo, etc.

Il processo Zygote 1/2

In fase di inizializzazione del sitema viene creato un processo denominato Zygote.

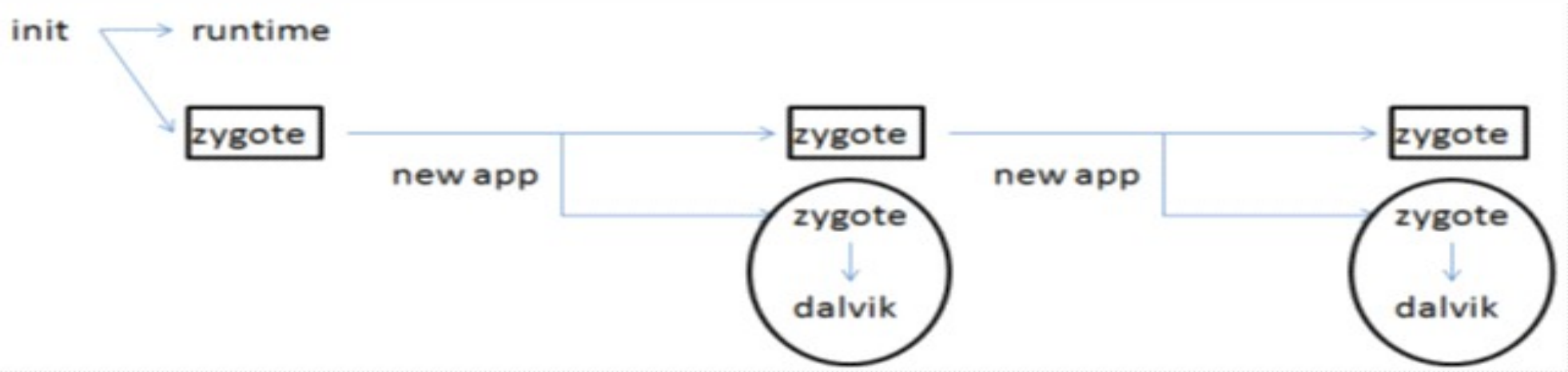
Esso si occupa di:

- Caricare le Core system libraries dex-files
- Inizializzare la Dalvik VM

Nel momento in cui viene avviata una Activity il processo Zygote subisce una fork e successivamente:

- Sono caricati i dex files specifici dell'Applicazione
- Vengono condivise le Core System Libraries

Il processo Zygote 2/2



Il processo Zygote rimane in esecuzione in background.
All'avvio di una nuova applicazione, il processo esegue una fork:
•uno dei due processi creati rimane in esecuzione in background
•l'altro inizializza un nuovo esemplare di DVM.

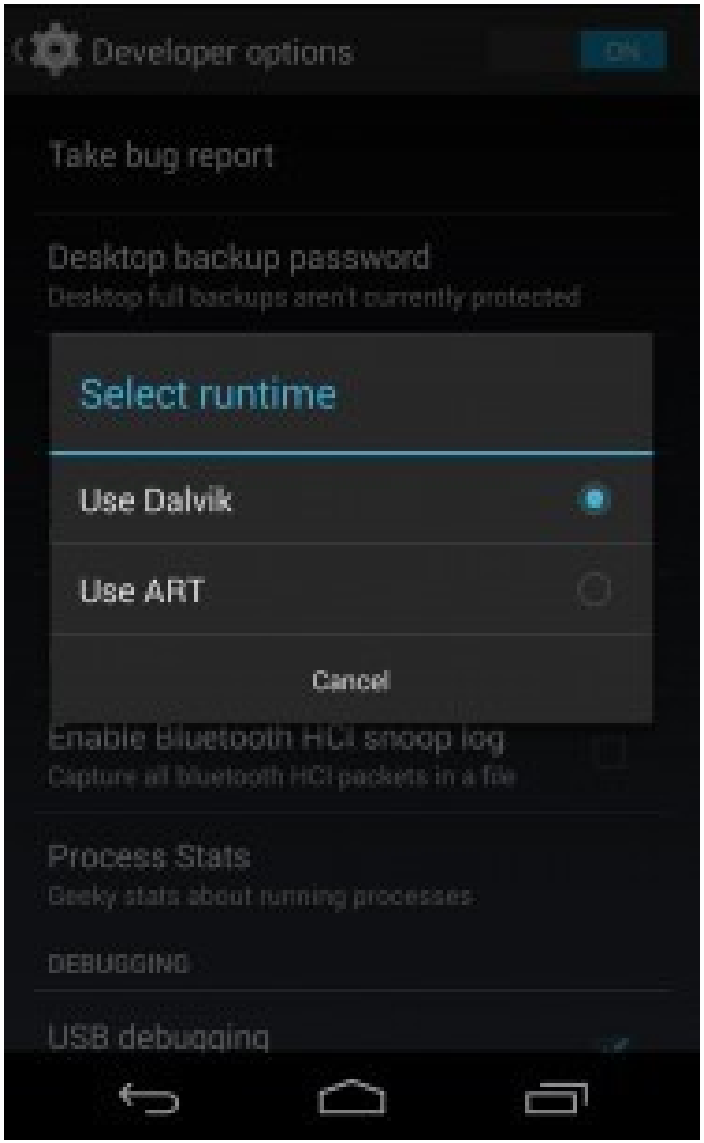
Se durante l'esecuzione di questa applicazione deve essere lanciata una ulteriore applicazione, il processo in background esegue una secondo fork.
In ogni momento è quindi disponibile un processo Zygote per l'inizializzazione di una nuova DVM.

Il nuovo Android Runtime 1/2

- Il problema maggiore del codice interpretato è legato al fatto che in esecuzione è più letto (spesso molto più lento) di un equivalente codice compilato nativamente in C/C++
- Per quanto detto in precedenza, l'efficienza della DVM nasconde bene questo limite
- In ogni caso l'esecuzione di codice interpretato comporta maggiore occupazione della CPU, conseguente riduzione della vita della batteria e della reattività complessiva

Il nuovo Android Runtime 2/2

- Per questa ragione da Google sta lavorando ad una nuova DVM, chiamata Android Runtime (ART) disponibile a partire dalla 4.4 in via sperimentale
- ART usa un metodo misto denominato ahead-of-time (AOT)
- Con abilitato ART, il codice dell'App viene compilato in codice nativo quando si installa la App.



Conclusioni

Anche se all'apparenza la macchina virtuale usata da Android può sembrare un clone di quella Java, ovvero uno stratagemma usato da Google per non pagare royalties a Sun Microsystems, in realtà è stato fatto un accurato lavoro di ottimizzazione per permettere che le applicazioni Android possano essere eseguite abbastanza velocemente anche su telefoni con risorse limitate, in particolare con poca RAM.