Android in Google I/O

- Google I/O で解説された Android の内部
  - Android 解剖
  - Dalvik の中身

サン電子株式会社
サンコミュニケーションズ事業部
水野光男

Disclaimer

- Official な情報ではありません。
  - Google I/O で行われたセッションの伝聞です。
  - 聞き間違い、翻訳の不備、勘違い、妄想が含まれる可能性があります。
- ソース
  - http://sites.google.com/site/io/
    - Anatomy & Physiology of an Android
    - Dalvik VM Internals

Google I/O

- 5/28, 29 @ San Francisco
- 世界中から 3,000 人の参加者
- Sessions, sessions, sessions
  - Google Web Toolkit
  - Gears
  - Google Maps
  - Google App Engine
  - Android

Contents

- Kernel の拡張
- Binder
- ボトルネックはバッテリー
- Bionic
- HAL
- Dalvik VM

Kernel の拡張

- http://git.android.com
- Alarm
- Ashmem – Shared Memory
- Binder
- Power Management
- Low Memory Killer
- Kernel Debugger
- Logger

Binder

- Why?
  - IPC はセキュリティが懸念される
  - Java シリアライゼーションはオーバーヘッドが大きい
- How?
  - Shared Memory 方式によるパフォーマンス改善
  - プロセス単位のスレッドプール
  - 同期 I/O コール

Power Management

- バッテリーが最大のボトルネック
- Linux PM の上に拡張
- アプリケーションからアクセス可能
  - FULL_WAKE_LOCK
    - CPU ON, LCD ON
  - PARTIAL_WAKE_LOCK
    - CPU ON, LCD may OFF
- ご利用は計画的に

Bionic

- カスタマイズされた libc
- 組み込み用に最適化
- Why?
  - BSD ライセンス
  - サイズ
  - 速度

Bionic featuring...

- pthread
- setprop/getprop
- logging

Bionic trade off...

- POSIX 完全準拠ではない。
  - C++ 例外
  - i18n
- glibc 互換ではない。
- ネイティブアプリケーションは Bionic にリンク。

HAL

- Hardware Abstruction Layer
- 上位層をハードウェアから分離
  - デバイスドライバを Apache 2.0 ライセンス化可能
  - pmap
    - persistent/physical mmap

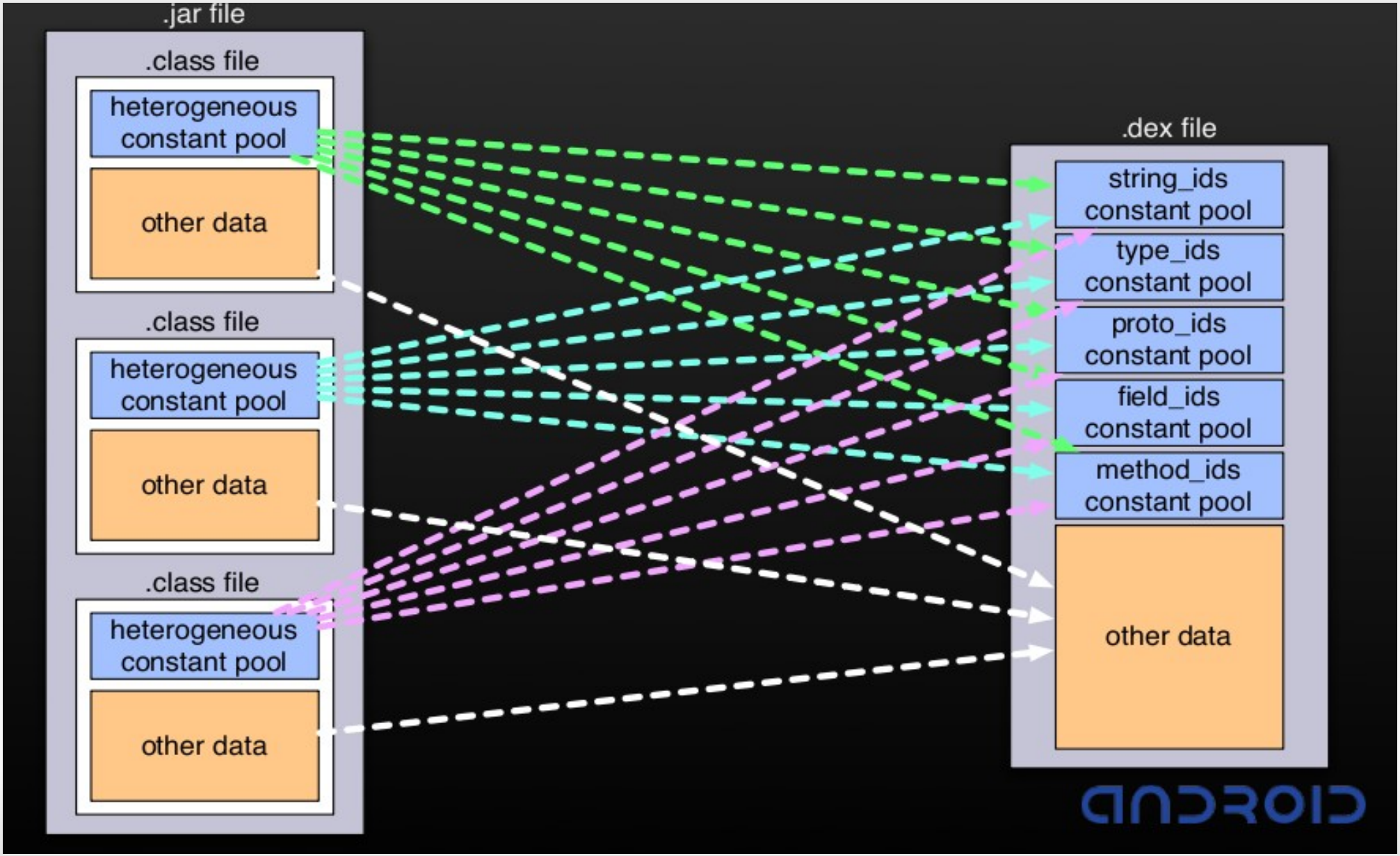
Dalvik, the Virtual Machine

- カスタム、クリーンルーム
- 組み込み用に最適化
  - 速度
    - 200 - 500MHz の CPU
  - メモリサイズ
    - RAM 64MB, ROM 64MB
    - アプリケーションは 20MB 使用可能
    - 10MB の巨大なシステムライブラリ
  - バッテリー

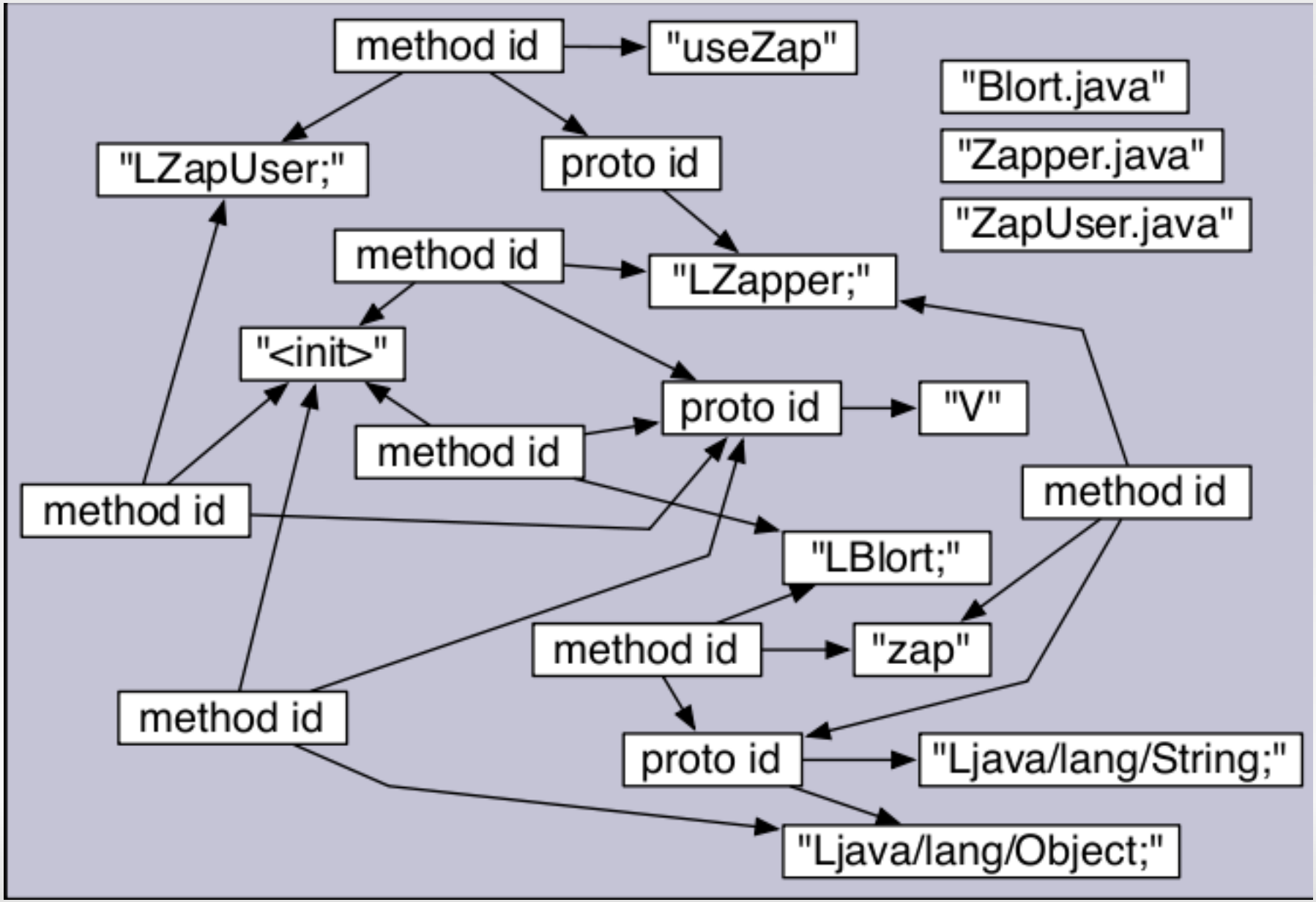
Dex ファイルの構造

- dx tool でコンパイル
  - header
  - string_ids　”Hello, World”
  - type_ids　int, String()
  - proto_ids　void fn(int), String fn()
  - field_ids　String.offset, Interger.MAX
  - method_ids　String.printf(), Array.size()
  - class_defs
  - data

Dex File Anatomy



Dex File Shared Constant Pool



Dex vs. Jar

	jar	zip	dex	(KB)
Web Browser	470	232	209	
Alarm Clock	119	61	53	

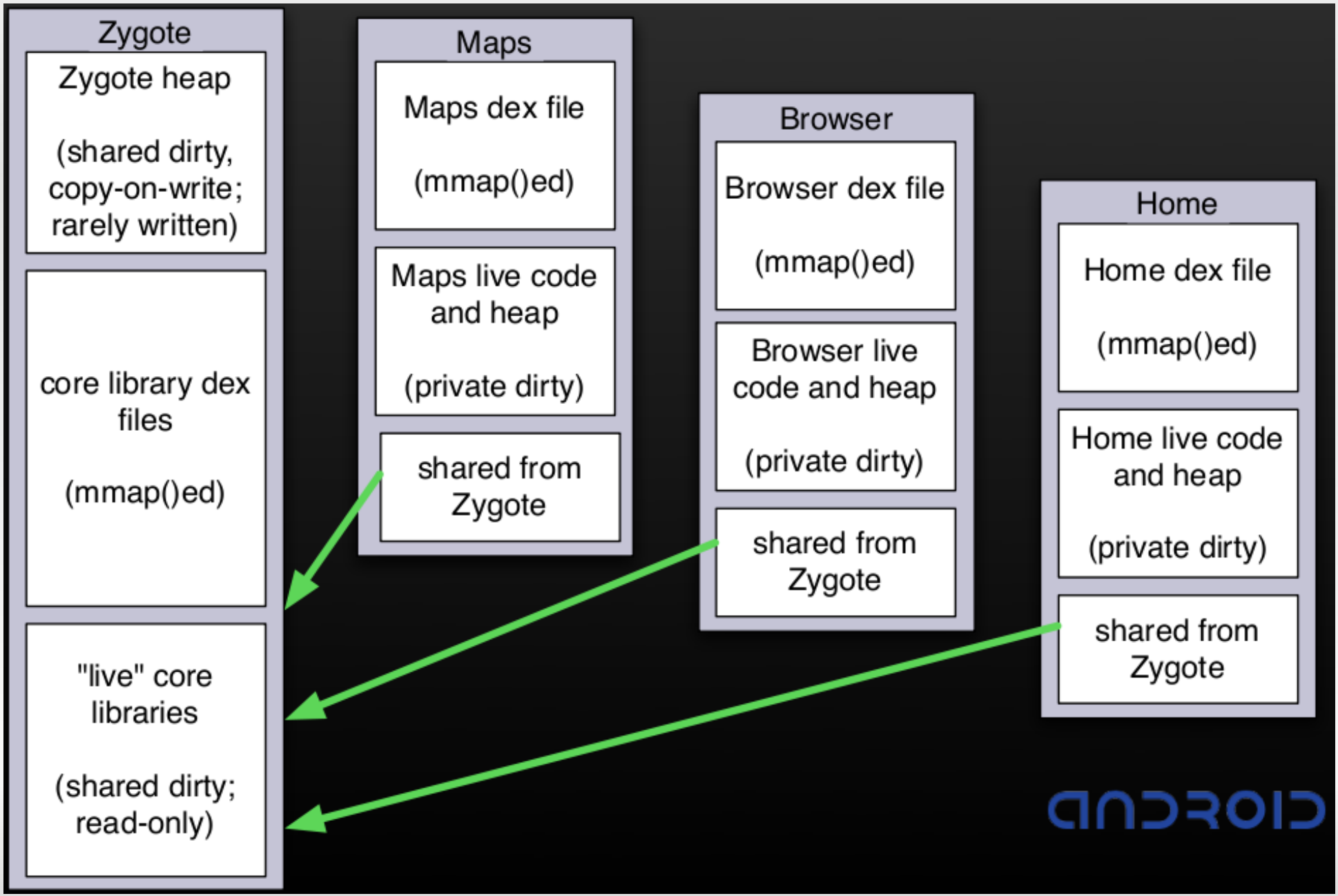
ファイルサイズ

- 圧縮しなくても jar の半分
- 圧縮した jar より小さい

メモリ使用の効率化

- クリーン
  - mmap, GC
- ダーティでプライベート
  - malloc，アプリケーションで管理
- ダーティでシェアード
  - Zygote（接合子）
    - VM で最初に起動
    - クラスをプリロードし初期化
    - システム全体で使用するダーティメモリをシェア

Zygote の導入


Zygote
Zygote heap
(shared dirty, copy-on-write; rarely written)
core library dex files
(mmap()ed)
"live" core libraries
(shared dirty; read-only)
Maps
Maps dex file
(mmap()ed)
Maps live code and heap
(private dirty)
shared from Zygote
Browser
Browser dex file
(mmap()ed)
Browser live code and heap
(private dirty)
shared from Zygote
Home
Home dex file
(mmap()ed)
Home live code and heap
(private dirty)
shared from Zygote
android

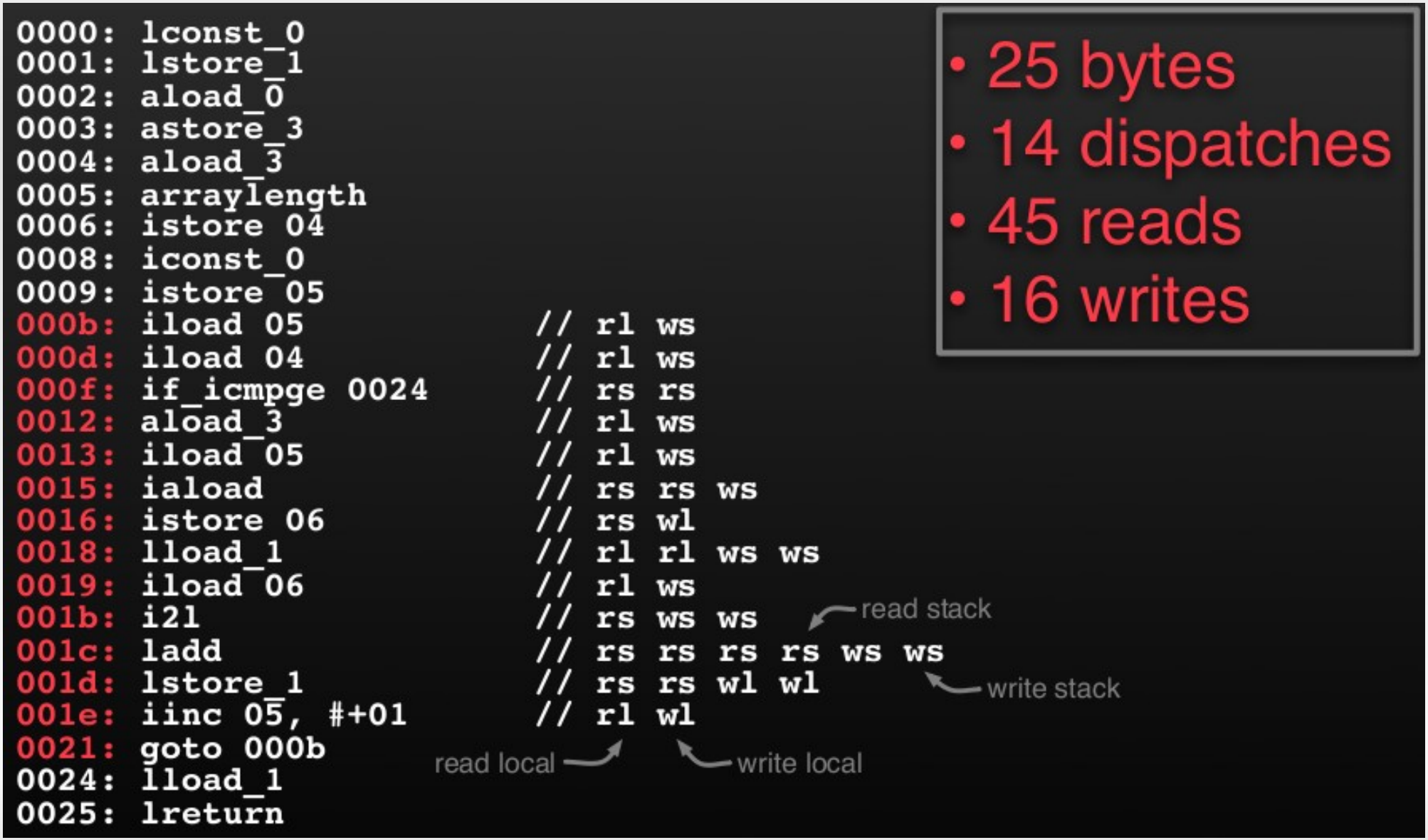
レジスタ vs. スタック

- Dalvik はレジスタマシン
  - 通常の Java VM はスタック
- インストラクション 30% 削減
- コードユニット 35% 削減
- 命令中のバイト数 35% 増加
  - 2 バイト単位で処理
  - 命令実行の効率化

実行効率改善例

```
public static int sumArray(int[] arr) {
    long num = 0;
    for (int i : arr) {
        sum += i;
    }
    return sum;
}
```

.class の場合


0000: lconst_0
0001: lstore_1
0002: aload_0
0003: astore_3
0004: aload_3
0005: arraylength
0006: istore 04
0008: iconst_0
0009: istore 05
000b: iload 05 // rl ws
000d: iload 04 // rl ws
000f: if_icmpge 0024 // rs rs
0012: aload_3 // rl ws
0013: iload 05 // rl ws
0015: iaload // rs rs ws
0016: istore 06 // rs wl
0018: lload_1 // rl rl ws ws
0019: iload 06 // rl ws
001b: i2l // rs ws ws
001c: ladd // rs rs rs rs ws ws
001d: lstore_1 // rs rs wl wl
001e: iinc 05, #+01 // rl wl
0021: goto 000b
0024: lload_1
0025: lreturn
read stack
write stack
read local
write local
• 25 bytes
• 14 dispatches
• 45 reads
• 16 writes

.dex の場合

```
0000: const-wide/16 v0, #long 0
0002: array-length v2, v8
0003: const/4 v3, #int 0
0004: move v7, v3
0005: move-wide v3, v0
0006: move v0, v7
0007: if-ge v0, v2, 0010                  // r r
0009: aget v1, v8, v0                     // r r w
000b: int-to-long v5, v1                  // r w w
000c: add-long/2addr v3, v5               // r r r r w w
000d: add-int/lit8 v0, v0, #int 1         // r w
000f: goto 0007
0010: return-wide v3
```

- 18 bytes
- 6 dispatches
- 19 reads
- 6 writes

改善例まとめ

	size	dispatch	read	write
.class	25	14	45	16
.dex	18	6	19	6

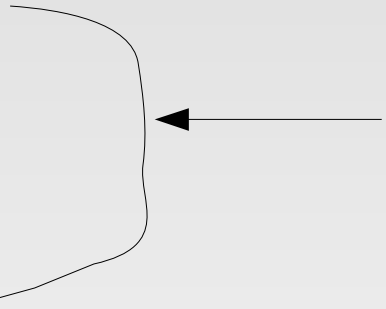
サイズは小さく
コードは高速化
メモリアクセスを削減

バッテリーが最大のボトルネック

- CPU やメモリのようには、バッテリーは進化しない
- 行儀のよいアプリケーション
  - Sleep – Wake -Sleep
  - WAKE_LOCK

賢くループ

(1) for (int i = initializer ; i >= 0 ; i--)
(2) int limit = calculate limit;
    for (int i = 0 ; i < limit ; i++)
(3) Type[] array = get array;
    for (Type obj : array)
(4) for (int i = 0 ; i < array.length ; i++)
(5) for (int i = 0 ; i < this.var ; i++)
(6) for (int i = 0 ; i < obj.size ; i++)
(7) Iterable<Type> list = get list;
    for (Type obj : list)



おすすめ

危険！危険！

Born to be Embedded

- CPU とメモリサイズはボトルネックではない
  - ファイルサイズ /Dex，メモリ /Zygote，速度 / レジスタ
- バッテリーがボトルネック
  - CPU やメモリのようには進化しない
  - 良く寝るアプリはよいアプリ
- Android は組み込み用として徹底的に考えられ設計されている。

Points of Interests

- http://code.google.com/intl/ja/android/
- http://sites.google.com/site/io/
- http://groups.google.co.jp/group/android-SDK-Japan
- http://groups.google.co.jp/group/android-developers-japan
- http://groups.google.co.jp/group/android-internals