MDCC 2015
2015 移动开发者大会
Mobile Developer Conference China 2015

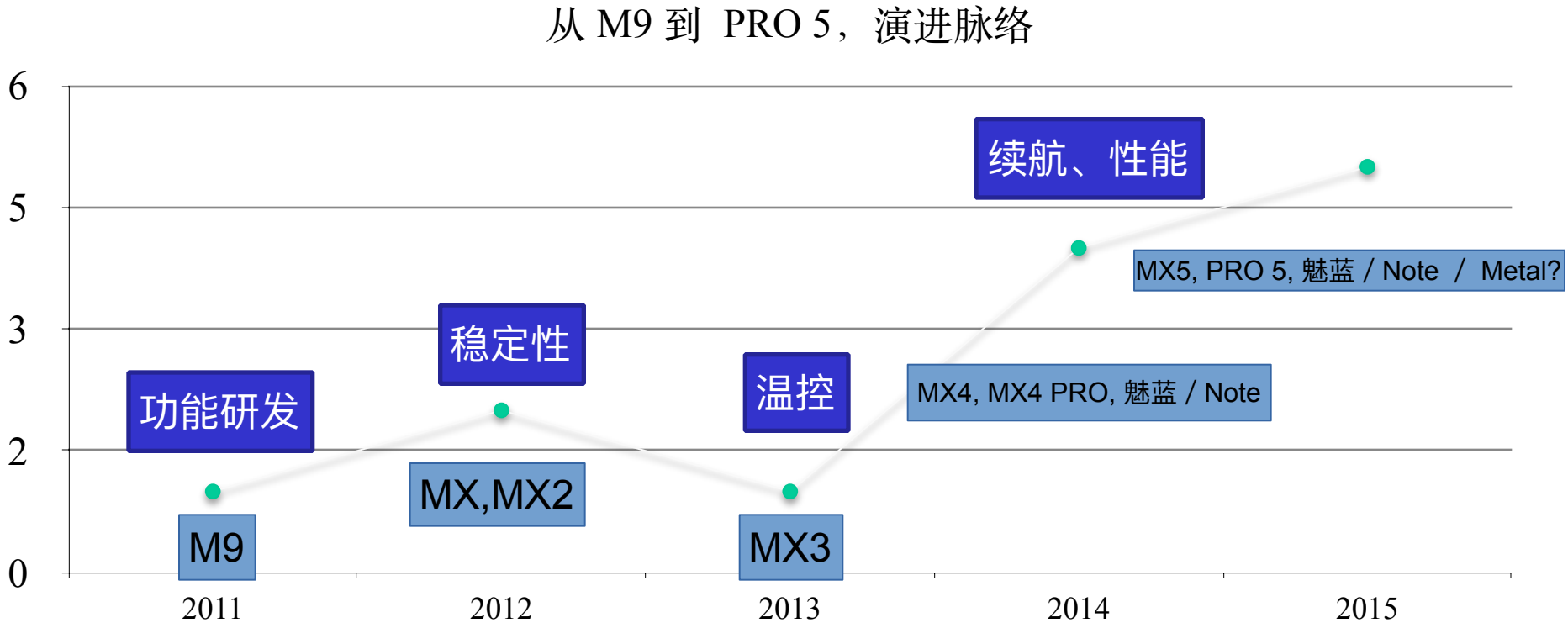
智能手机底层系统优化的演进
—— 从 M9 到 PRO 5

吴章金 @ 魅族科技
wuzhangjin@gmail.com
http://tinylab.org

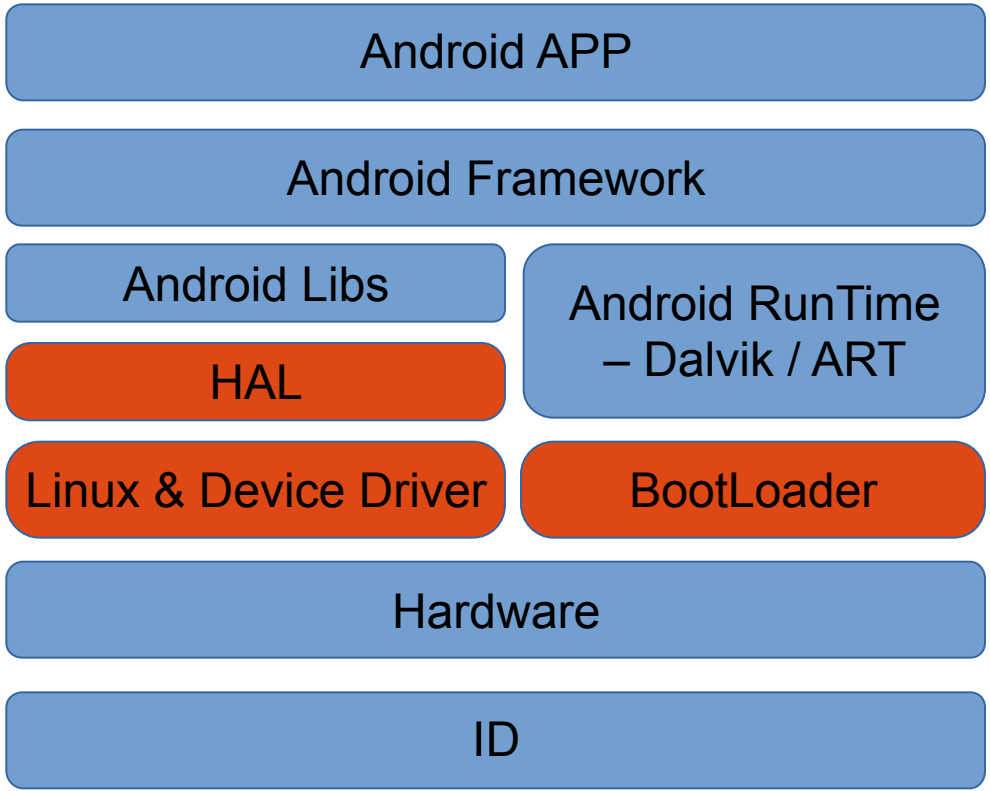
演示概要

- 相关背景介绍
  - 背景简介：从 M9 到 PRO 5
  - 关注对象；优化目标；演进过程
- 底层系统优化
  - 4 大项：稳定性、温控、续航、性能
  - 5 方面：问题、难点、目标、措施、技术
- 开放问题讨论
  - 团队架构、研发流程、标准建设、行业协作

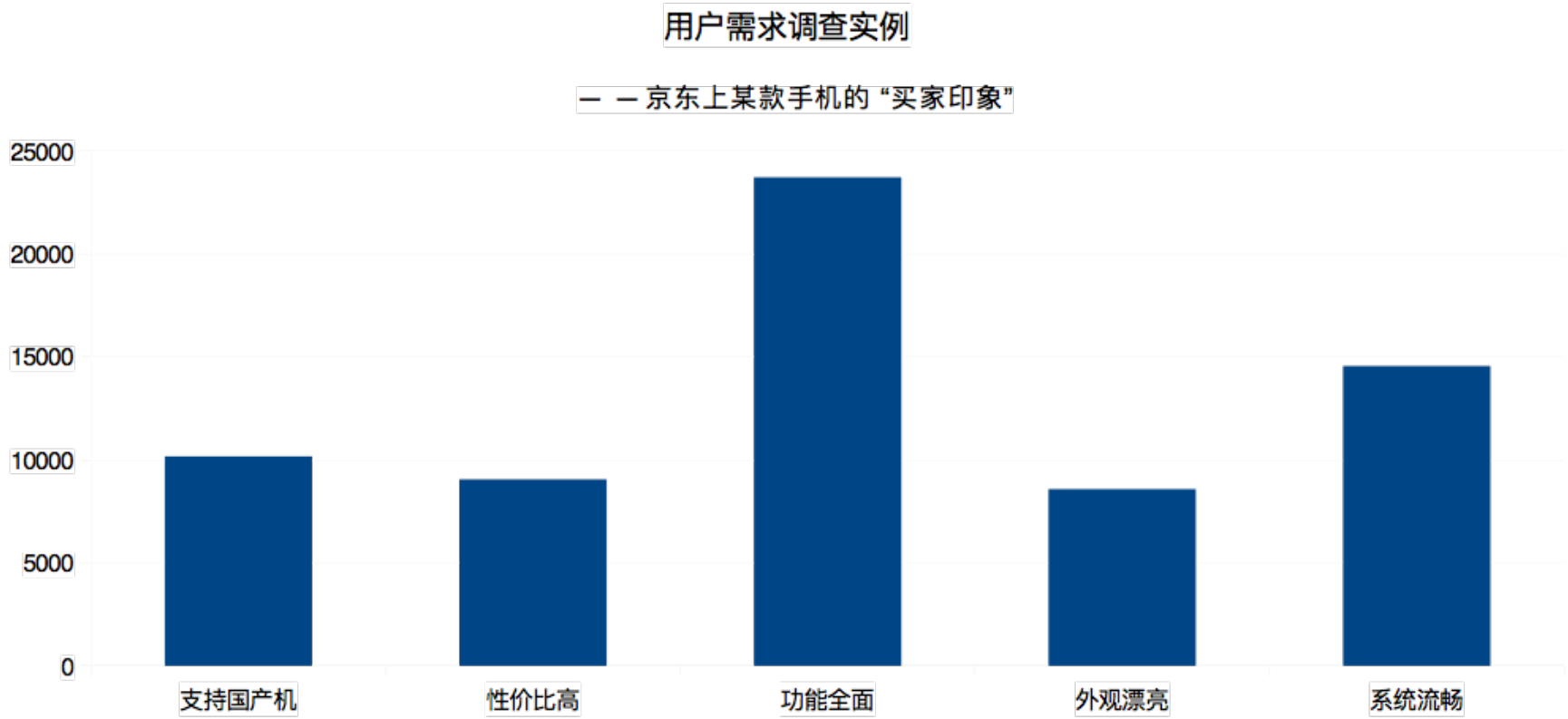
背景介绍


从 M9 到 PRO 5，演进脉络
6
5
3
2
0
功能研发
M9
稳定性
MX,MX2
温控
MX3
续航、性能
MX4, MX4 PRO, 魅蓝 / Note
MX5, PRO 5, 魅蓝 / Note / Metal?
2011
2012
2013
2014
2015

关注对象


Android APP
Android Framework
Android Libs
HAL
Android RunTime
– Dalvik / ART
Linux & Device Driver
BootLoader
Hardware
ID

优化目标(1)


用户需求调查实例
— —京东上某款手机的“买家印象”
25000
20000
15000
10000
5000
0
支持国产机
性价比高
功能全面
外观漂亮
系统流畅

优化目标(2)

- 需求分析（从用户体验的角度）
  - 听觉（音质）
  - 视觉（外观）
  - 触觉（温度）
  - 其他（时间？私密？安全？）
    - 稳定
    - 流畅
    - 续航
    - 安全

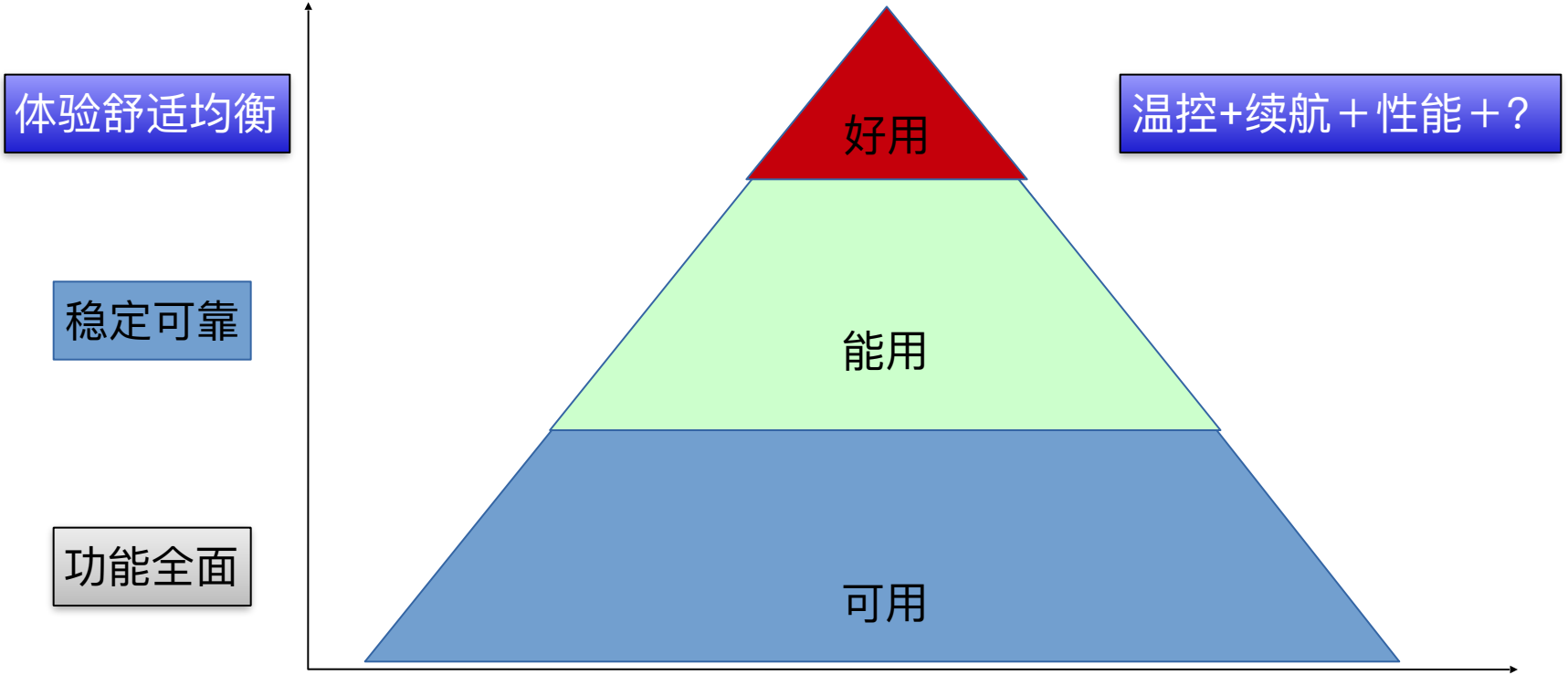
温控优化

稳定性优化

性能优化

续航优化

演进方向

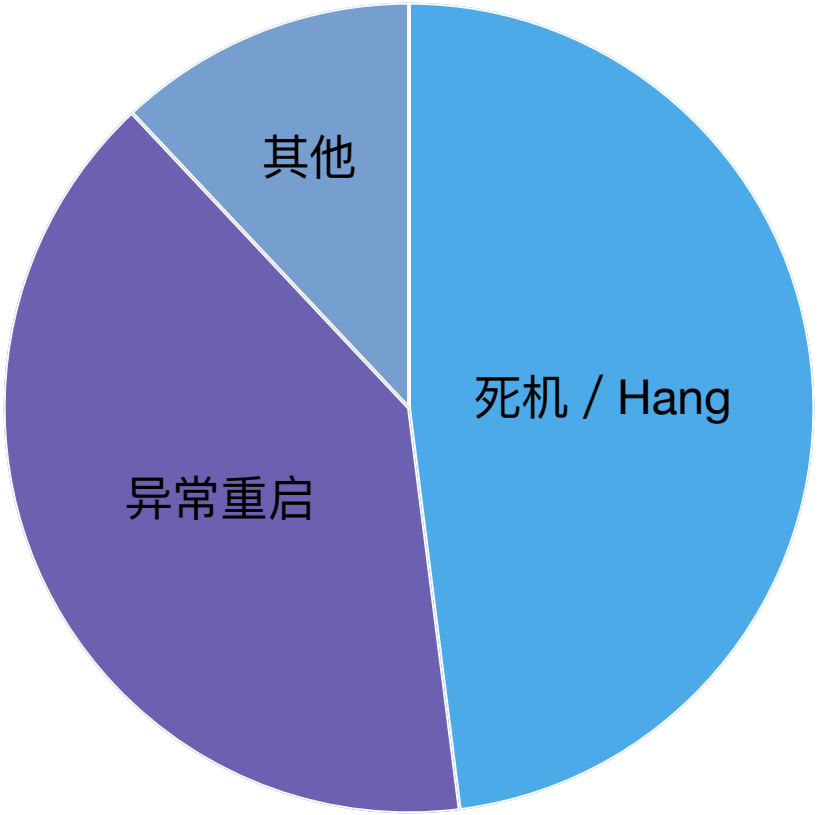


智能手机系统研发三大重要阶段

功能研发

- 满足用户的基本需求
- 研发流程的必要阶段
  - 先有功能，后有优化
- 功能机 → 智能机 的必要过渡期
  - 硬件技术的革新 → Touch, Fingerprint
  - 系统创新与开放 → Android
  - 人才与技术积累 → 硬件/Android/Linux

稳定性：问题

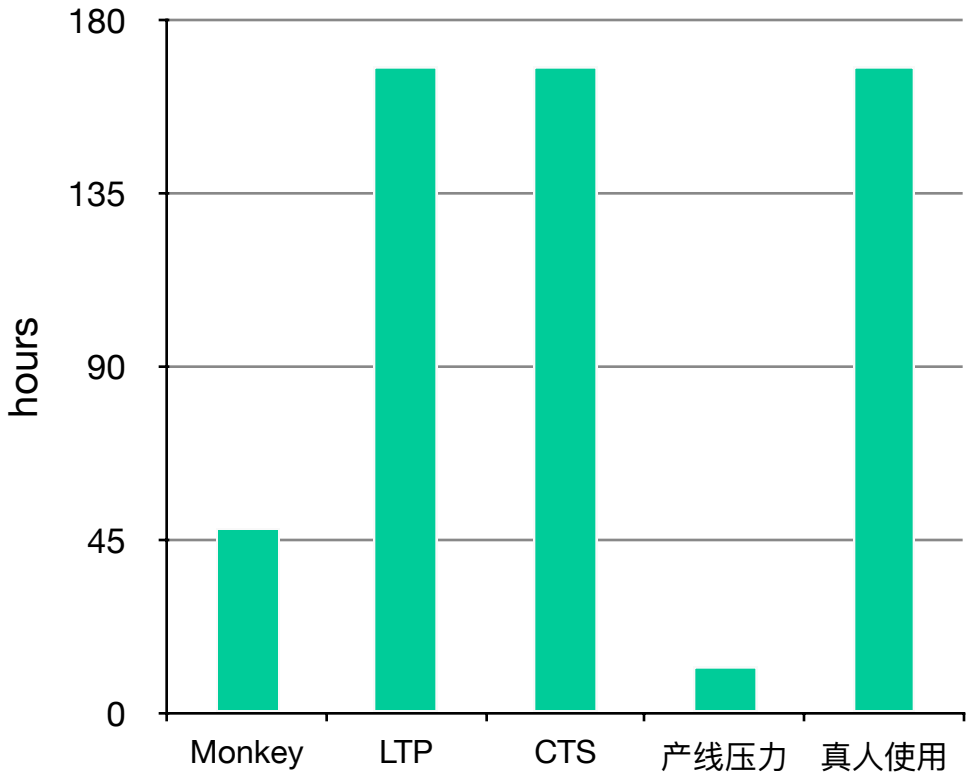


稳定性：难点

- 原因复杂多样
- 调试、跟踪困难
- 技术要求广度和深度
- 与硬件或固件密切相关，依赖供应商支持
- 跨多个部门，涉及采购与生产环节

稳定性：目标

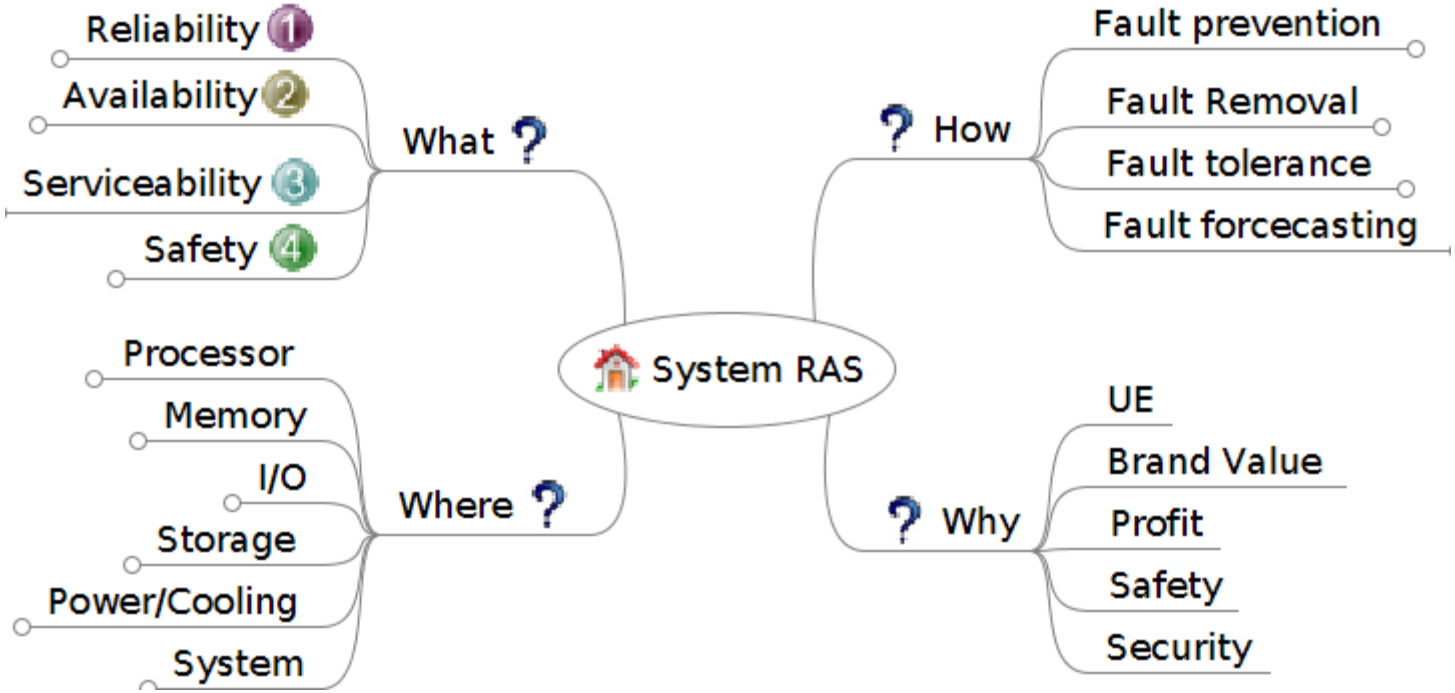
- 各类服务持续可用
  - 不死机
- 可较快恢复服务
  - 死机后能恢复
  - Fail stop → Fail Restart



稳定性：措施

- BugFix → 兵来将挡
- Faq总结 → 举一反三
- 技术预研 → 各个击破
- 组建团队 → 专业专注
- 加强流程 → 风险管控
- 系统架构 → 全面突破

稳定性：RAS建模



系统稳定性研究

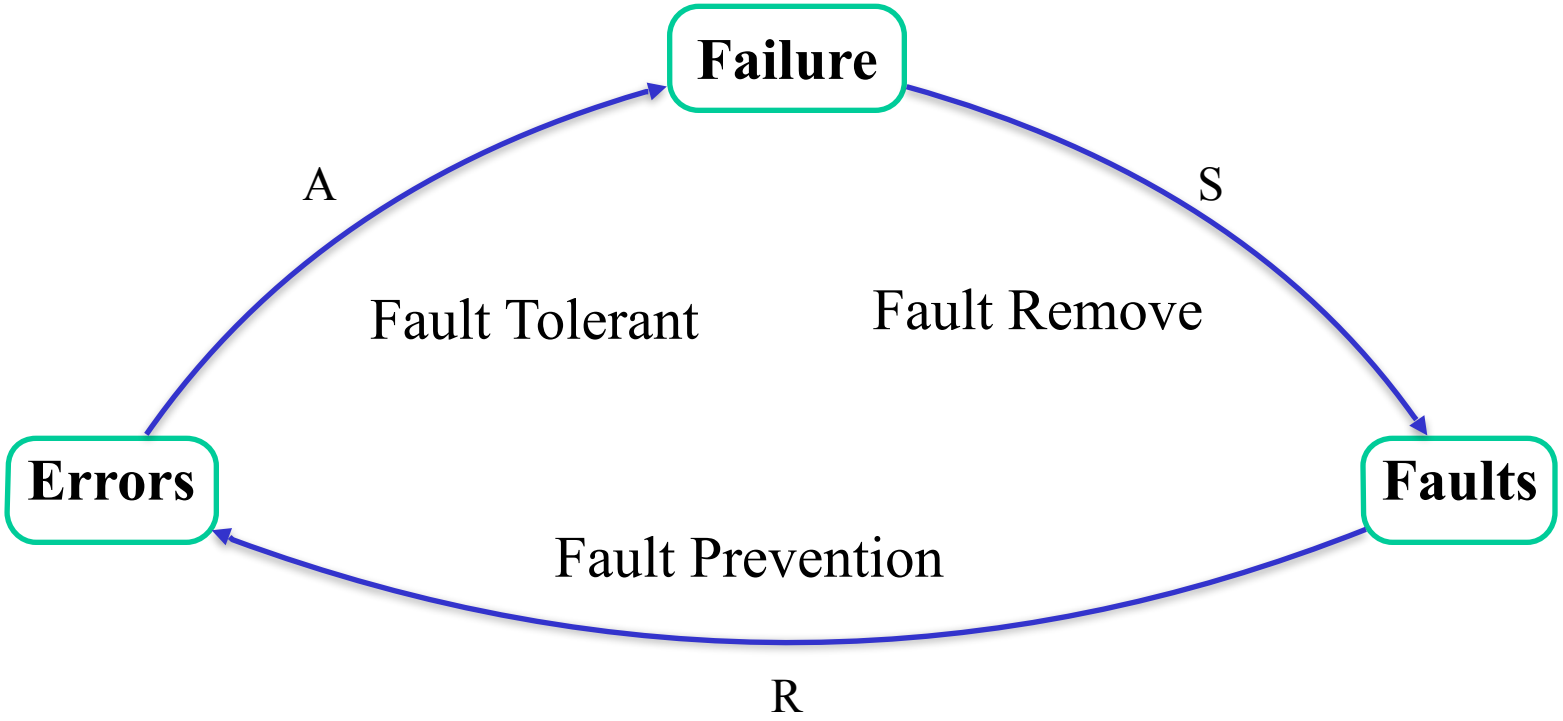
稳定性：RAS概念（1）

- R: Reliability / 可靠性
  - 系统正常运行的能力
  - E.g. 一周停机几次，一天死机几次
- A: Availability /可用性
  - 能否使用，即使不能完全正常运行
  - E.g. 停机一次有多久，一个礼拜有多少时间不能正常使用
- S: Serviceability / 可服务性,可维修性
  - 系统恢复正常运行需要的时间和难度
  - E.g. 出现问题后有多快能恢复，恢复难度有多大

稳定性：RAS概念（2）

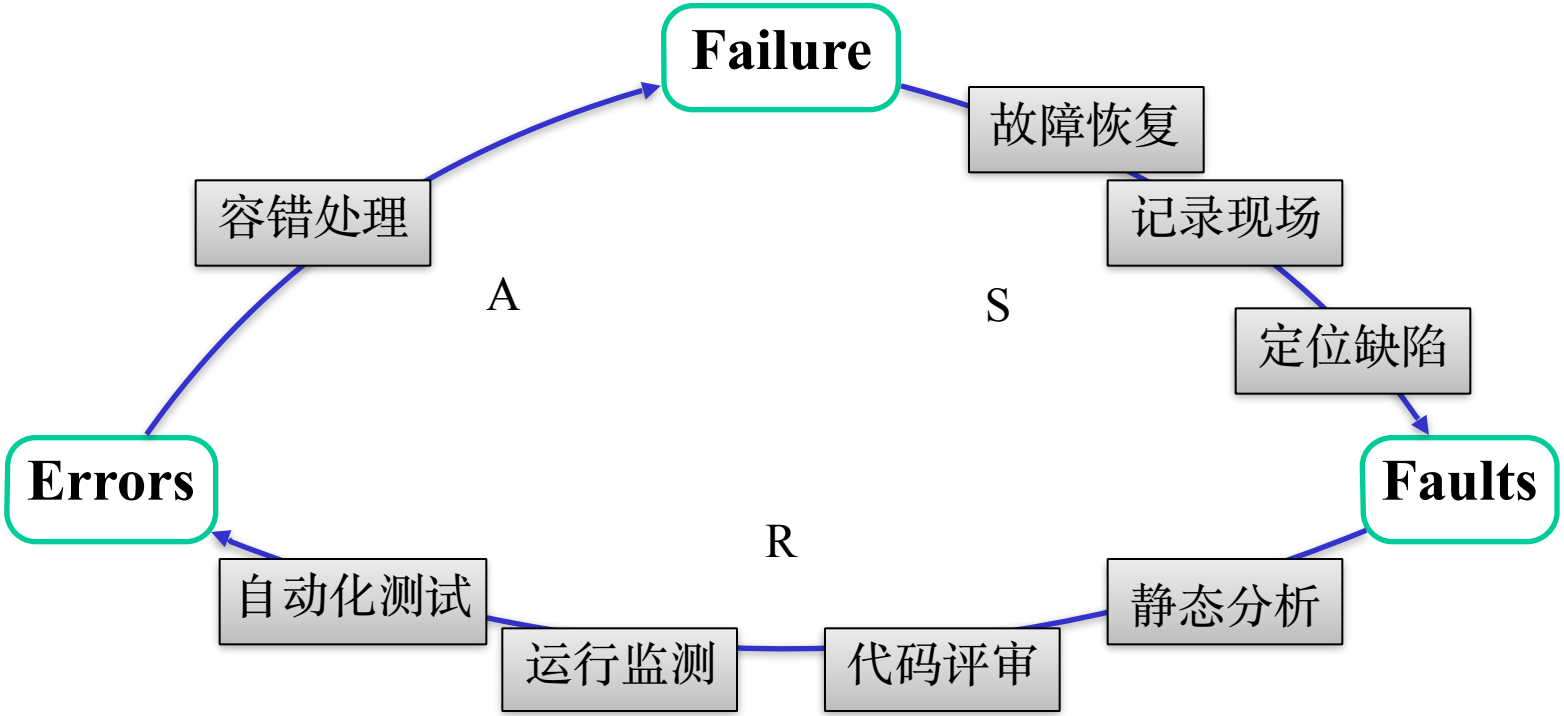
- Faults / 缺陷
  - 软件中固有的不足和瑕疵
  - 需求不明确、设计缺陷、算法漏洞、代码不规范、硬件老化等
- Errors / 错误
  - 软件运行过程中偏离目标的不正确状态(正确性、准确度)
  - 上述缺陷引起的的偏离正确状态的各种问题，例如固件加载失败
- Failures / 故障
  - 系统中表现出了与需求不一致的行为
  - 上述错误导致的偏离客户需求的各种系统表现
  - 例如：Modem 固件多次加载失败后不能打电话

稳定性：RAS架构（1）



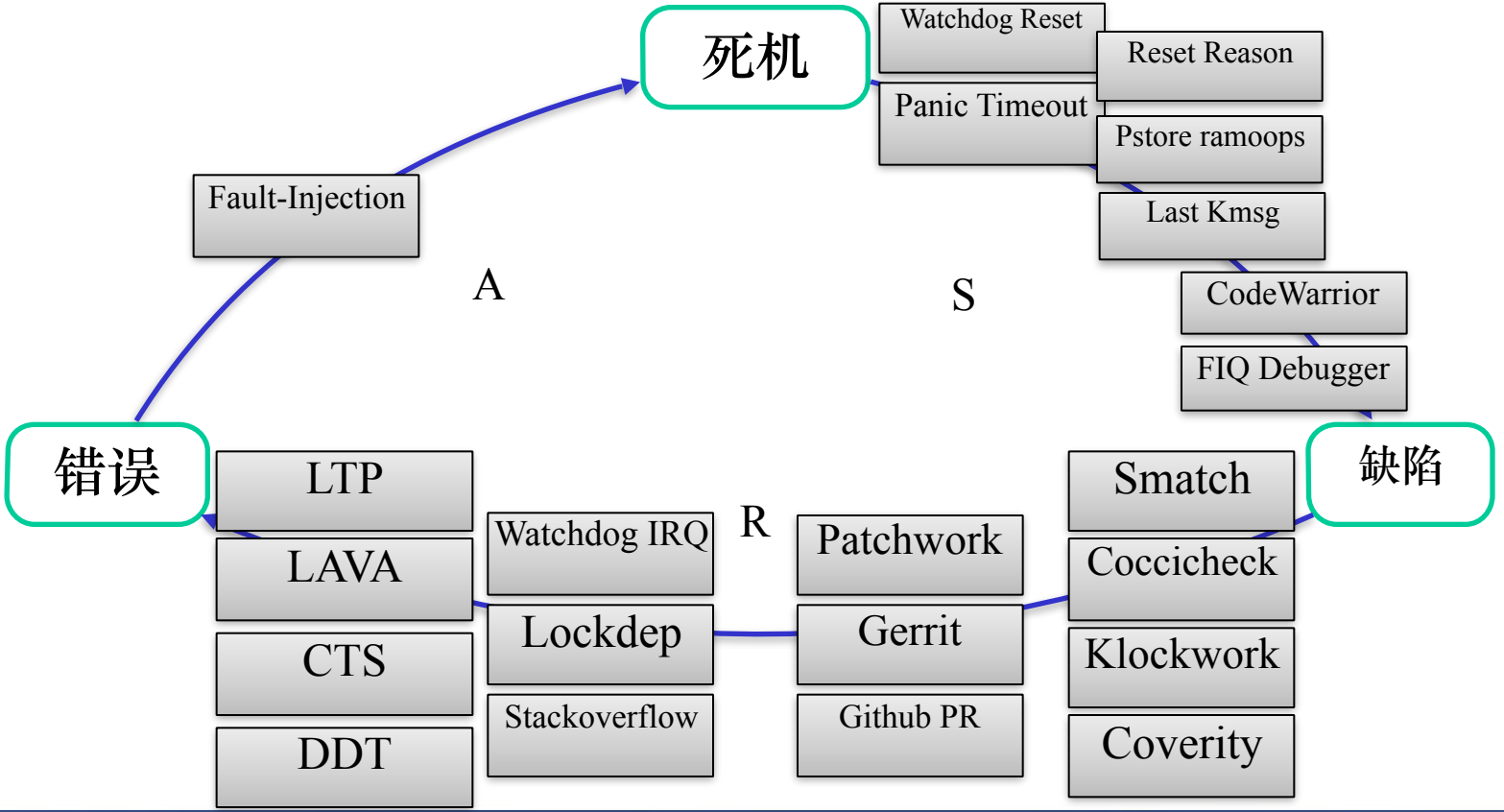
系统稳定性架构

稳定性：RAS架构（2）

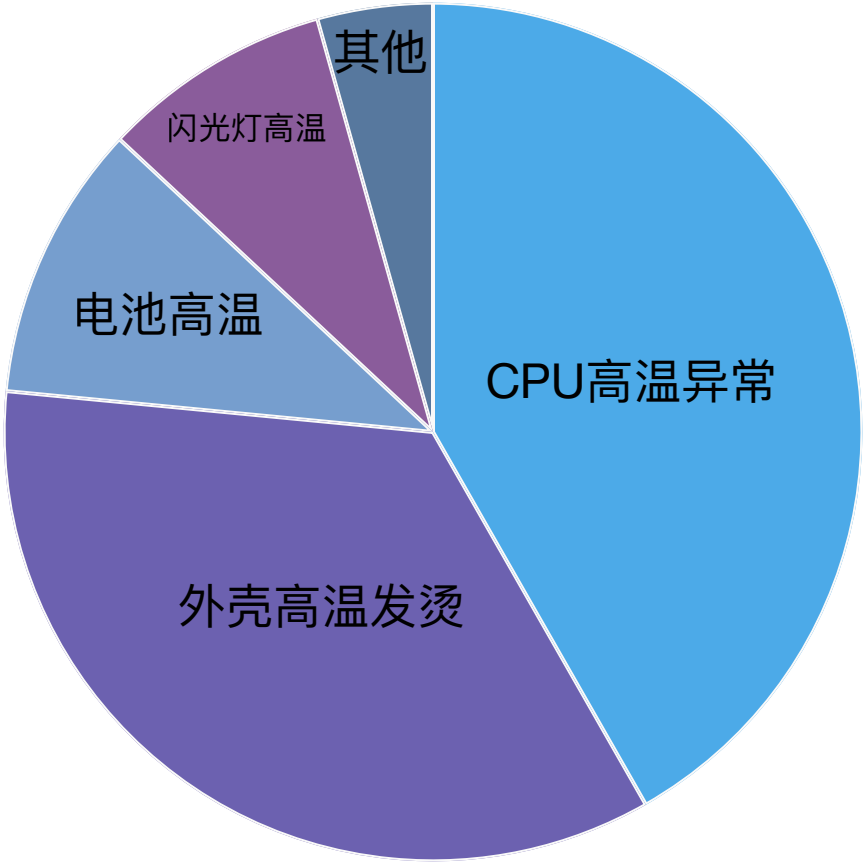


系统稳定性物理架构

稳定性：技术


死机
Watchdog Reset
Reset Reason
Panic Timeout
Pstore ramoops
Last Kmsg
Fault-Injection
A
S
CodeWarrior
FIQ Debugger
缺陷
错误
LTP
LAVA
CTS
DDT
Watchdog IRQ
Lockdep
Stackoverflow
R
Patchwork
Gerrit
Github PR
Smatch
Coccicheck
Klockwork
Coverity

温控：问题

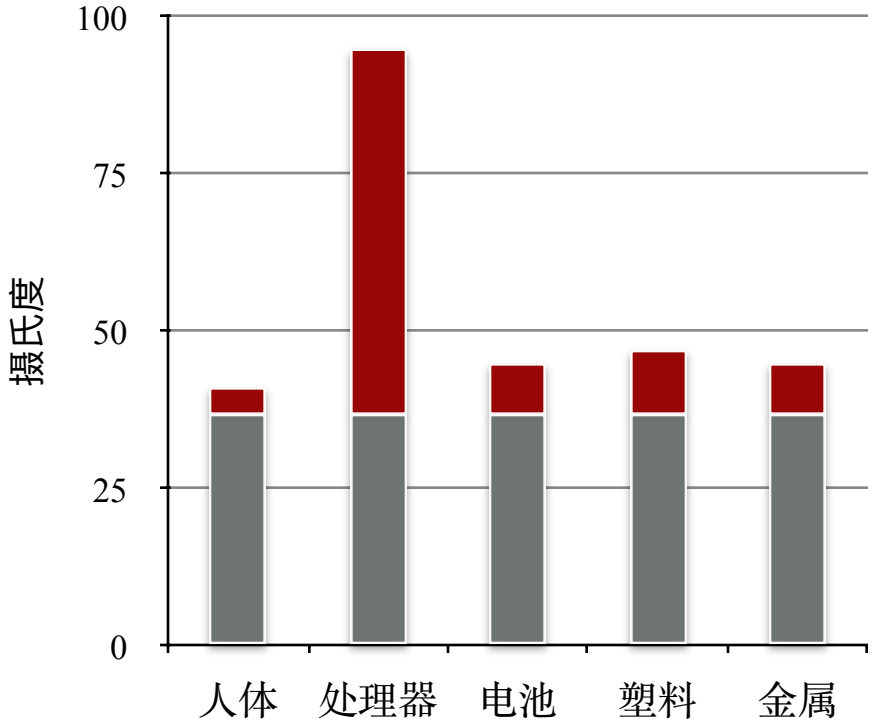


温控：难点

- 与硬件、工艺密切相关
  - 严重依赖各类器件、工艺，**A15 v.s. A53**
- 与结构密切相关
  - 散热、隔热技术以及成本，**温差控制**
- 与性能密切相关
  - 简单限制资源必然降低性能、甚至影响功能

温控：目标

- 器件安全
  - CPU：65 ~ 95
  - 电池：45
- 人体安全
  - 人体：39~41 高烧
- 人体触觉舒适度
  - 金属：45、塑料：47



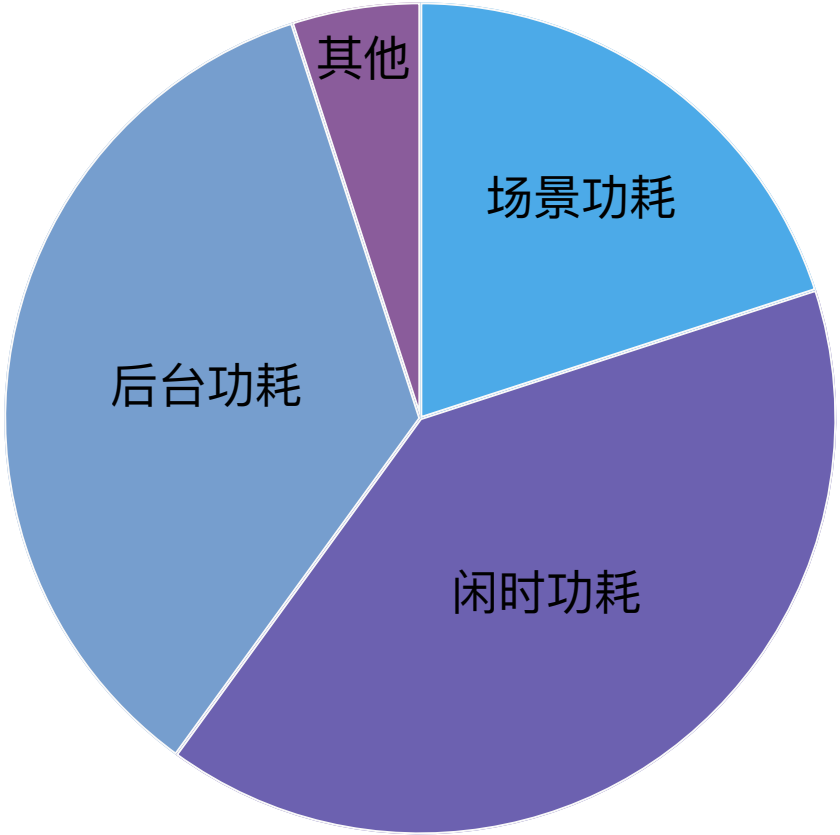
温控：措施

- 温度传感器
  - CPU、外壳、电池、闪光灯等
- 资源限制
  - 频率、核数、LP状态、亮度
- 协同工作
  - 升级器件工艺
  - 功能优化（充电、相机等）
  - 结构散热、隔热优化 → 控制处理器与外壳温差
  - 性能优化

温控：技术

- 温度采集
  - Hwmon
- 资源控制
  - Thermal / Cooling Device
  - **PM Qos**
- 兼顾性能
  - **IPA**
- 温度测量
  - 红外温度计；热成像仪

续航：问题

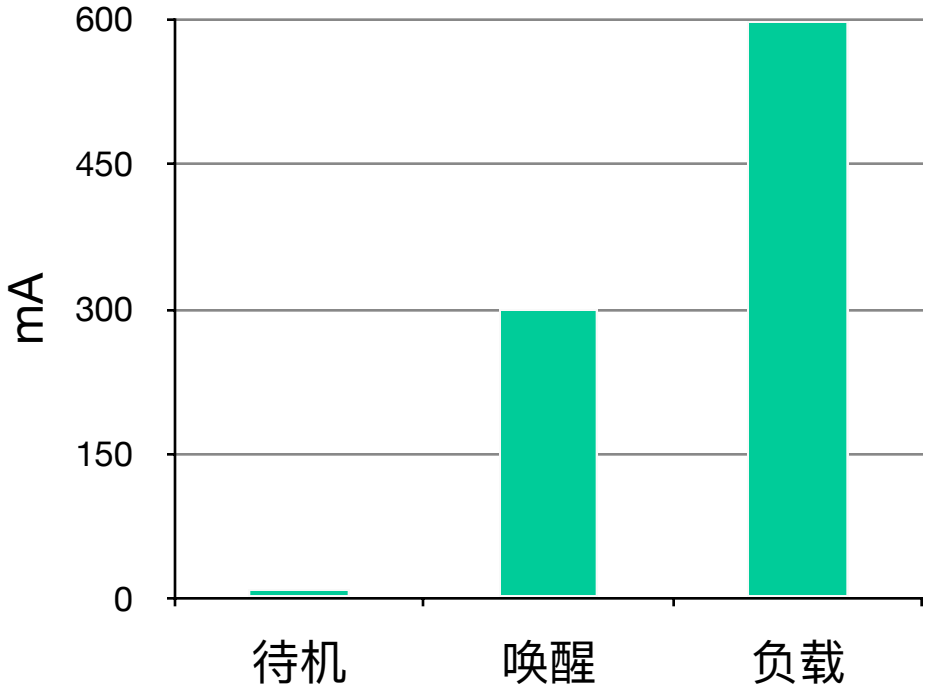


续航：难点

- 与性能密切相关
  - 如简单限制资源必然降低性能
- 与设计、制程和工艺密切相关
  - A72 v.s. A53
  - 28nm v.s 14nm
  - LP v.s. HPM
- 与系统和应用开发规范密切相关
  - 自启动、后台服务泛滥，**系统厂商不作为 v.s. 应用厂商懈怠**
  - 消息推送、Alarm API 等规范缺失，**Power Tail**

续航：目标

- 延长各场景使用时间
  - E.g. 移动入库标准
- 闲时不浪费
  - 所有未操作场景
- 后台不乱跑
  - 用户未主动启动服务



续航：措施

- 场景功耗
  - 采用更节能工艺和器件
  - 细化场景目标、并针对性优化
  - 节能模式分级，按需限制资源利用
- 闲时功耗
  - 自启动管理
  - 智能后台管理
  - Power Tail 管理：网络推送、ALARM对齐
  - Ondemand 服务启动

续航：技术（1）

- 基本技术
  - System Suspend，即 **STR** → 兼顾性能与续航的典范
  - Autosuspend + Wakelock
  - Clock Gating
  - PMIC Regulator
  - DVFS，包括 CPUFreq, BUSFreq，MEMFreq…
  - CPUIdle
  - CPUHotplug
  - Runtime PM
  - 采用专有硬件：HWC v.s. GPU
  - 配合硬件技术：例如：OLED，屏幕自刷新
  - 图形显示优化：降 FPS，例如：动画设计帧率优化

续航：技术（2）

- CPUIdle
  - Tickless → Full Tickless
  - Dynamic IRQ Affinity → 中断送 timer 最先到期的 CPU
  - Power Efficient Workqueue → 允许调度器把 Workqueue 放到 non-idle CPU
- 多核管理
  - CPUQuiesce → 替换 CPU Hotplug，兼顾性能与稳定性
- 硬件资源管理
  - **PM Qos**
  - **Cgroup → CPUset**
  - **EAS → Power Aware Scheduler**
- 服务/APP管理
  - **Doze & APP Standby；绿色守护**
  - **Systemd?**

续航：技术（3）

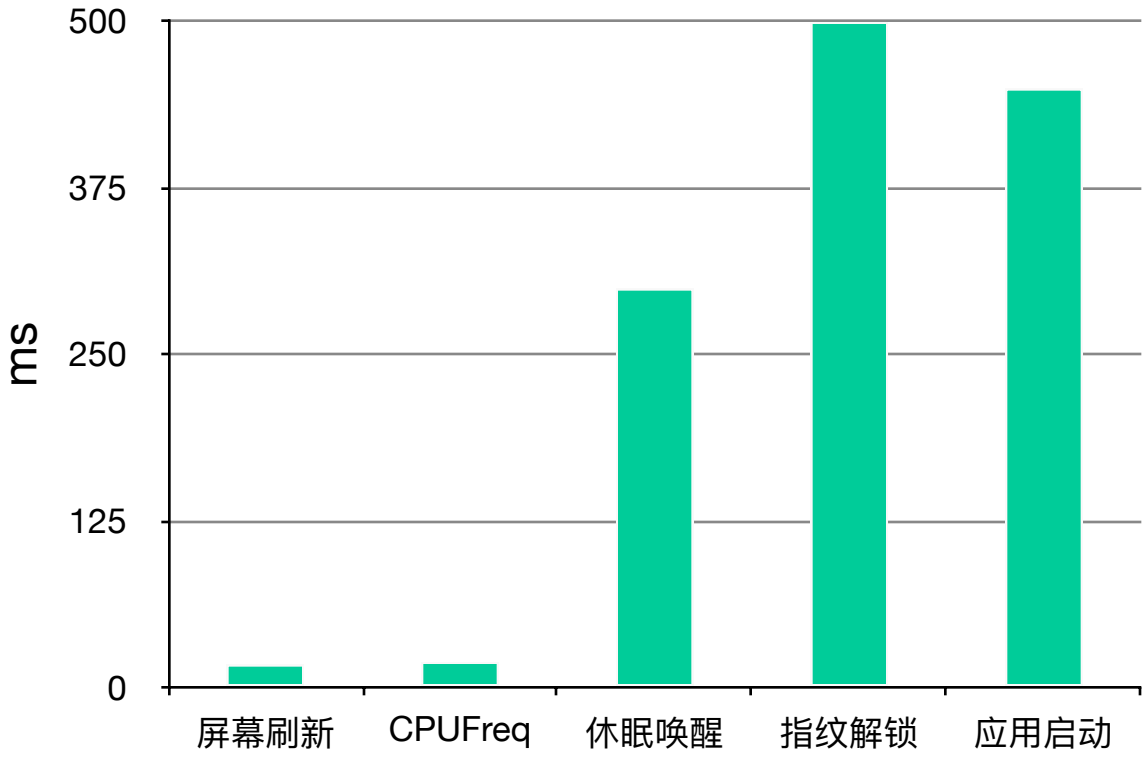
- 唤醒统计
  - PowerTop （Idle）
  - Wakeup Reason （STR）
  - BetterBatteryStats（WakeLock）
- 功耗测量
  - 支持 GPIB 的程控电源
- 功耗分析
  - **Oscilloscope** → 软件数字示波器

性能：问题

- 常规问题
  - 启动、唤醒、传输、读写慢等
- 异常卡慢
  - 拍照久了以后卡
  - 游戏部分场景丢帧

性能：目标

- 降低各场景时间开销
- 解决异常卡慢



性能：措施

- 正面分析各场景
  - 满足开发规范
  - 热点路径排查
- 满足资源要求
  - 升级硬件性能
  - 频率、核数等资源按需调配
- 异常卡慢
  - 内存泄漏排查
  - 低内存管理
  - 内存管理器配置排查

性能：技术（1）

- 常规技术
  - 遵守开发规范，例如
    - 消除 过度绘制
    - 正确使用 mdelay, msleep, usleep_range
  - 优化热点路径，Systrace, Perf, Ftrace, CyclicTest
- 资源保障
  - DVFS 调优（轮询）＋ PM Qos, Cgroup 即时请求（事件）
  - 内存泄漏排查，Kmemleak, leakcanary
  - 算法优化：Load Based v.s. 大数据 Based

性能：技术（2）

- 性能测量
  - grabserial （Bootloader）
  - scripts/bootgraph.pl （Linux）
  - AnalyzeSuspend （Suspend）
  - GPU 呈现模式 （GPU）
- 性能分析
  - **Oscilloscope** → 自动抓取热点数据

演示总结

- 演进过程
  - 功能研发 → 系统优化
  - 可用 → 能用 → 好用
- 系统优化
  - 4 大项：稳定性、温控、续航与性能
  - 5 方面：问题、难点、目标、措施、技术

开放问题（1）

- 团队架构
  - 项目为中心，还是以技术为中心?
  - **项目进度 v.s. 技术积累**
- 流程优化
  - 跨部门问题分析与协作过程?
  - **Top Down v.s. Down Top**
- 标准建设
  - 各项需求、用例、目标、步骤的标准化?
  - **行业标准 v.s. 用户需求**

开放问题（2）

- 技术突破
  - 扩大合作，与同行、社区、供应商等的合作?
  - **自研 v.s. 合作**
- 加强交流
  - 与用户的交流；各类人才的交流?
  - **封闭 v.s. 交流**

Thank You!

扫码关注 **泰晓科技**，嵌入式 Android/Linux 原创与交流！http://tinylab.org