输入法性能优化计划

1. 版本问题数据总览

8.2时期测试

测试场景	百度输入法华为版 V8.2.6.0	搜狗输入法 V8.19	讯飞输入法 V8.0.6589	优化目标	方案
26键汉字候选词响应	159	133	216	持平搜狗	各种气泡统一到相同浮层，8.3测试与搜狗一致
热启动	464	372	381	持平搜狗	分析之后与搜狗一致
拼音9键切符号面板	266	248	161	超过搜狗，争取持平讯飞	修改方案，从浮层改为面板，减少浮层启动时间
拼音9键切数字面板	210	171	146	持平搜狗	待进一步分析，8.3测试与搜狗一致
冷启动（切换输入法）	877	850	无	超越搜狗	重写Serviceloader增加冷启动速度，皮肤文件预编译+MMAP（默认皮肤硬编码？），硬键盘等待耗时，减少系统远程调用，SharedPreference加载优化，分dex优化
删除				保持	理论上和搜狗只是误差，和讯飞的差异需要进

上屏字符	170	173	149	优于搜狗	一步分析，策略上存在明显差异，例如数据统计，Sug，云输入等获取光标前内容
常驻内存	98.51	109.33	70.7	保持优于搜狗	不让AR等功能引入新问题
使用态CPU	3.03%	3.28%	2.70%	保持优于搜狗	不让输入引入CPU问题

8.3时期测试

测试场景	百度输入法 V8.3.0.5	搜狗输入法 V8.24	讯飞输入法 V8.1.7623	优化目标	方案
空规格内存	67.61	84.97	90.31	保持优于搜狗	AR SO加载时机优化，内核词库加载优化，ServiceLoader加载优化
开机内存	43.21	37.59	45.74	持平搜狗	重写Serviceloader降低内存，全面排查内存占用情况，分dex优化
空闲态CPU	0.05%	0.01%	0.00%	持平搜狗	待进一步分析
热启动	433	414	386	持平搜狗	分析之后与搜狗一致
9键中文候选	151	133	168	保持优于搜狗，不差于之前版本	分析之后与之前版本一致
26键英文候选	159	157	142	持平搜狗	待进一步分析
收面板	319	316	293	持平搜狗	待进一步分析

2. 问题分析与优化方案

2.1. 热启动

结论：与竞品相当，优化空间小，优化难度中，低优

QA测试case下的热启动流程：

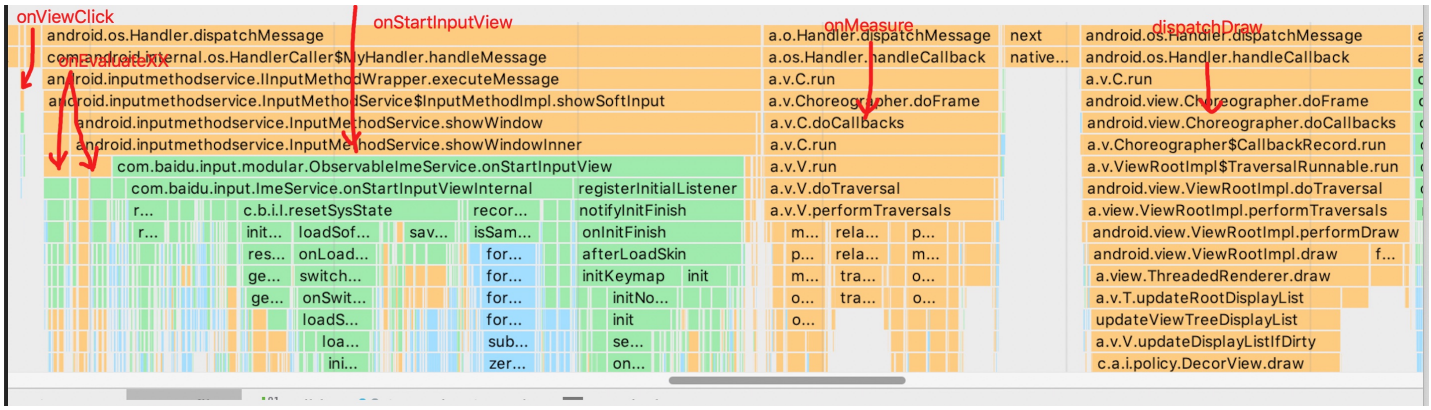
1.用户点击 -> 2.onViewClick响应 -> 3.onEvaluateXX -> 4.onStartInputView -> 5.onMeasure -> 6.dispatchDraw -> 7.开始起面板 -> 8.面板完全展现

QA计算时间为1-8总时间，其中属于输入法可以控制的逻辑为2-7，1-2与7-8的时间由系统决定。其中7-8是系统固定动画耗时，所有输入法理论值固定且相等，如有误差均为测试误差。

下图是QA最新测试结果的一个详细数据，可见总耗时与竞品的差距并不大，统计1-7耗时，我们的输入法比搜狗输入法低**8ms**（1-7耗时横向对比最精确，参考意义最大）。统计2-7耗时，我们的输入法比搜狗输入法低17ms（2-7耗时统计并不一定准确，参考意义低）。

测试项	百度输入法8.2.1.33	百度输入法V8.3.0.5	搜狗输入法V8.19	搜狗输入法V8.24
1-8总耗时	429ms	430ms	423ms	427ms
1-7总耗时	无数据	296ms	无数据	304ms
2-7总耗时	无数据	54ms	无数据	71ms

在同样环境下我们通过打点的方式得到2-7的数据在60-100ms左右，下图给出于2-7中的耗时详细分析，主要的耗时点在onStartInputView调用中，onMeasure基本是系统耗时，dispatchDraw中系统耗时与我们代码耗时占用相当。



而一次起面板中onStartInputView中的耗时占用基本在20-50ms之间，可做优化项如下：

1. recordUserAction延迟到onWindowShown调用，可优化2-5ms
2. 使用MMKV替代sharedpreference、Option优化性能，可优化2-3ms
3. SmartReply.checkIsSwitchOn调用逻辑优化，可优化0-1ms
4. 使用其他方案替代getScaledEmojiImg，可优化0-1ms
5. 统一权限检查，可优化1-5ms

这些优化项对整体耗时优化的力度有限，并不能起到很大作用。

2.2. 冷启动（切换输入法）

结论：与竞品差距小，优化空间大，优化难度高，高优

QA测试case下的冷启动的主要流程如下：

1.点击切换输入法，前一个输入法收起 -> 2.加载输入法APK -> 3.Application初始化 -> 4.ImeService.onCreate -> 5.ImeService.onInitializeInterface -> 6.ImeService.onStartInput -> 7.ImeService.onStartInputView -> 8.onMeasure - > 9.dispatchDraw -> 10.开始起面板 -> 11.面板完全展现

QA计算时间为1-11总时间，其中属于输入法可以控制的逻辑为2-10，1-2与10-11的时间由系统决定。其中10-11是系统固定动画耗时，所有输入法理论值固定且相等，如有误差均为测试误差，**比较特殊的是搜狗输入法关闭了冷启动的动画，因此不存在10-11的耗时**。

下图是QA最新测试结果的一个详细数据，可见统计总耗时我们比竞品高56ms，统计1-10耗时，我们的输入法比搜狗输入法**低81ms**（1-10耗时横向对比最精确，参考意义最大）。

测试项	百度输入法8.2.1.33	百度输入法V8.3.0.5	搜狗输入法V8.19	搜狗输入法V8.24
1-11总耗时	1024ms	1165ms	无数据	1109ms
1-10总耗时		X		X

在同样环境下我们通过打点的方式得到3-10的数据在Xms左右，下图给出该耗时详细分析，主要的耗时点有。

// TODO

这个流程里我们确定可以进行优化的项有：

1. ServiceLoader加载耗时优化，可优化50-100ms
2. 皮肤相关资源加载优化，默认皮肤进行hardcode，可优化？ms
3. app信息数据收集延迟，可优化10ms

可以优化但需要进一步技术调研的项有：

1. 优化1-2的耗时，dex分包，主dex轻量化处理，可优化？ms

这些优化项可以有效降低冷启动耗时，但是优化难度很大，需要较长时间处理。

2.3. 空规格内存

2.4. 开机内存

2.5. 收面板

2.6. 空闲态CPU

2.7. 9键中文候选

结论：与8.2版本一致，不用优化

用8.3.0.10灰度正式版本与8.2线上正式版本进行对比。把点击操作分成down与up两个部分，进行两部分数值累加，作为整体的点按时间。

1. 8.3.0.10版本 down的平均值是24.72ms,up的平均值是58.909ms
2. 8.2线上版本 down的平均值是22.6ms，up的平均值是54.4ms

从两者的分析流程来看，流程也是相同的，两个版本并没有明显时间以及流程上的差异。QA的策略结果，我们认为是在误差范围内。并且QA的测试方法，其实是把从点按到没有抬手的过程中，停留的时间也被记录进去，这部分停留没有抬手的时间，也会造成总的时间增大。

2.7. 26键中文候选

2.8. 拼音9键切符号面板

2.9. 删除上屏字符

2.10. 26键英文候选

3. 排期规划

4. 其他规划

性能排查工具规划

1. Code段内存排查工具，基于smaps，帮助排查目前code段内存占用比重较大的问题
2. Native内存排查工具，帮助排查native内存问题
3. 内存自动测试脚本

崩溃与卡顿问题

Flywheel3.0

1. 强化短阈值卡顿监控能力
2. 增加系统ANR收集，由于卡顿监控方案存在一定性能损耗，release版本上会改用系统ANR收集
3. 卡顿/ANR/崩溃上传机制优化，提升数据回传率
4. 重写native崩溃收集，提升崩溃收集能力
5. 效果数据埋点（回传率、收集率）

线上崩溃卡顿

1. 线上卡顿（卡顿率降低到1%以下）
2. 崩溃（崩溃率降低到0.1%以下）

ShowX监控平台

1. Showx与icafe打通，展示优化（服务器端工作）
2. mapping文件反扰码支持（客户端提供脚本）
3. 堆栈聚类支持（客户端提供脚本）

编译期优化

1. 分dex方案：提升首次启动速度，降低首次启动内存
2. proguard优化：降低包大小
3. resource.arsc扰码：降低包大小

包大小过大

1. 表情采用静默下发；AR相关功能，静默下发；内置词库格式优化等
2. 屏幕适配方案优化：去除不必要资源文件