iOS输入法启动速度优化

百度手机输入法
范敏虎

目录

- 产品简介
- 面临问题
- 解决方案

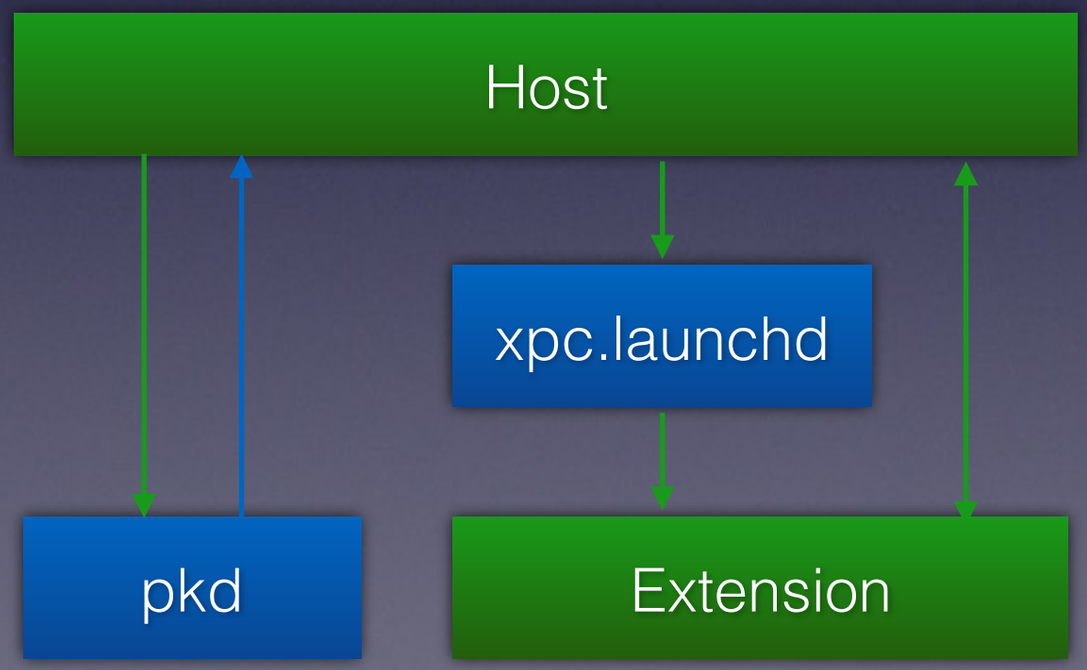
产品介绍

- 产品简介
  - 百度输入法iOS版本是2014年苹果在iOS8上开放Extension开发上线到AppStore的,前身是百度输入法越狱版
- Extension
  - 通过XPC被HostApp调起
  - 生命周期不同于普通app
- Extension限制
  - 启动时间限制
  - 内存限制

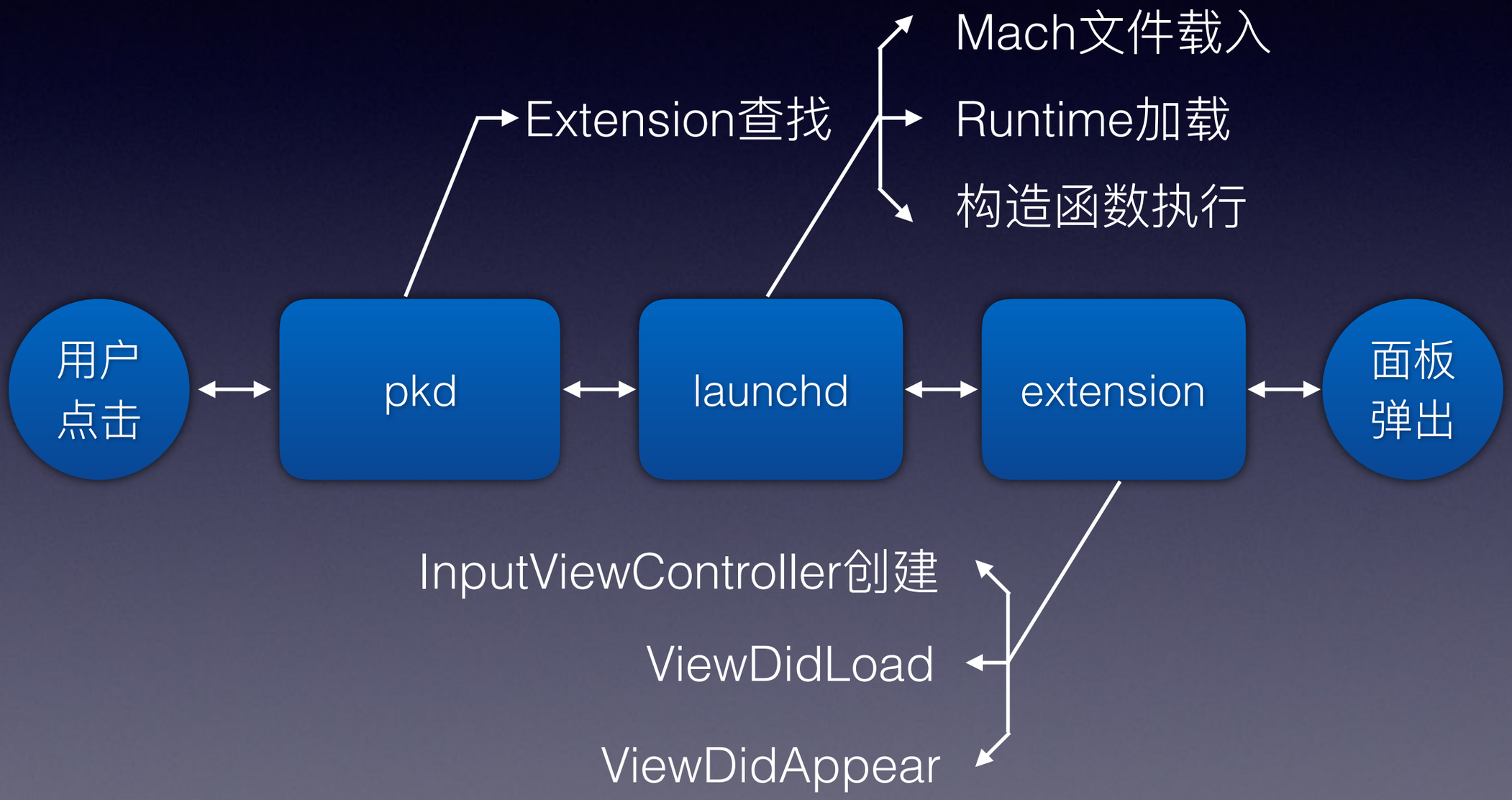
Bai du 百度

启动过程

- Extension查找
  - Host通过xpc的方式请求pkd,找到需要的Extension
- Extension启动
  - Host通过name连接Extension
  - xpc.launchd启动Extension
- Host与Extension交互
  - Host调用Extension展示键盘



启动过程


Extension查找
Mach文件载入
Runtime加载
构造函数执行
用户点击
pkd
launchd
extension
面板弹出
InputViewControlIer创建
ViewDidLoad
ViewDidAppear

面临的问题

- 皮肤文件解码载入需要大量CPU时间
- UI渲染需要一定的CPU时间
- 内核词库的载入耗时了大量I/O时间
- Extension要处理一些静默任务

解决方案

- 将皮肤生成时对图片预先解码
- 将面板对象缓存在缓冲池中
- 内核载入与查询都在独立线程异步执行
- 静默任务在低优先级队列单独调度

皮肤编/解码

- 皮肤由Ini&CSS&图片切片构成
- 对Ini 及 CSS 进行解码并缓存
- 对图片进行切分,解码,并缓存

皮肤编/解码

- NSDictionary

```
+ (NSDictionary *)dictionaryWithIniFile:(NSString *)filePath {
    NSString* contents = [NSString stringWithContentsOfFile:filePath encoding:NSUTF8StringEncoding error:nil];
    NSArray*  lines    = [contents componentsSeparatedByCharactersInSet:[NSCharacterSet newlineCharacterSet]];
    NSMutableDictionary* dict = [NSMutableDictionary dictionaryWithCapacity:10];
    for (NSString* line in lines) {
        NSString* item = [line stringByTrimmingCharactersInSet:[NSCharacterSet whitespaceCharacterSet]];
        if (isSection) {
            // Add a new section
        } else {
            // Add key/value to section
        }
    }
    return dict;
}
```

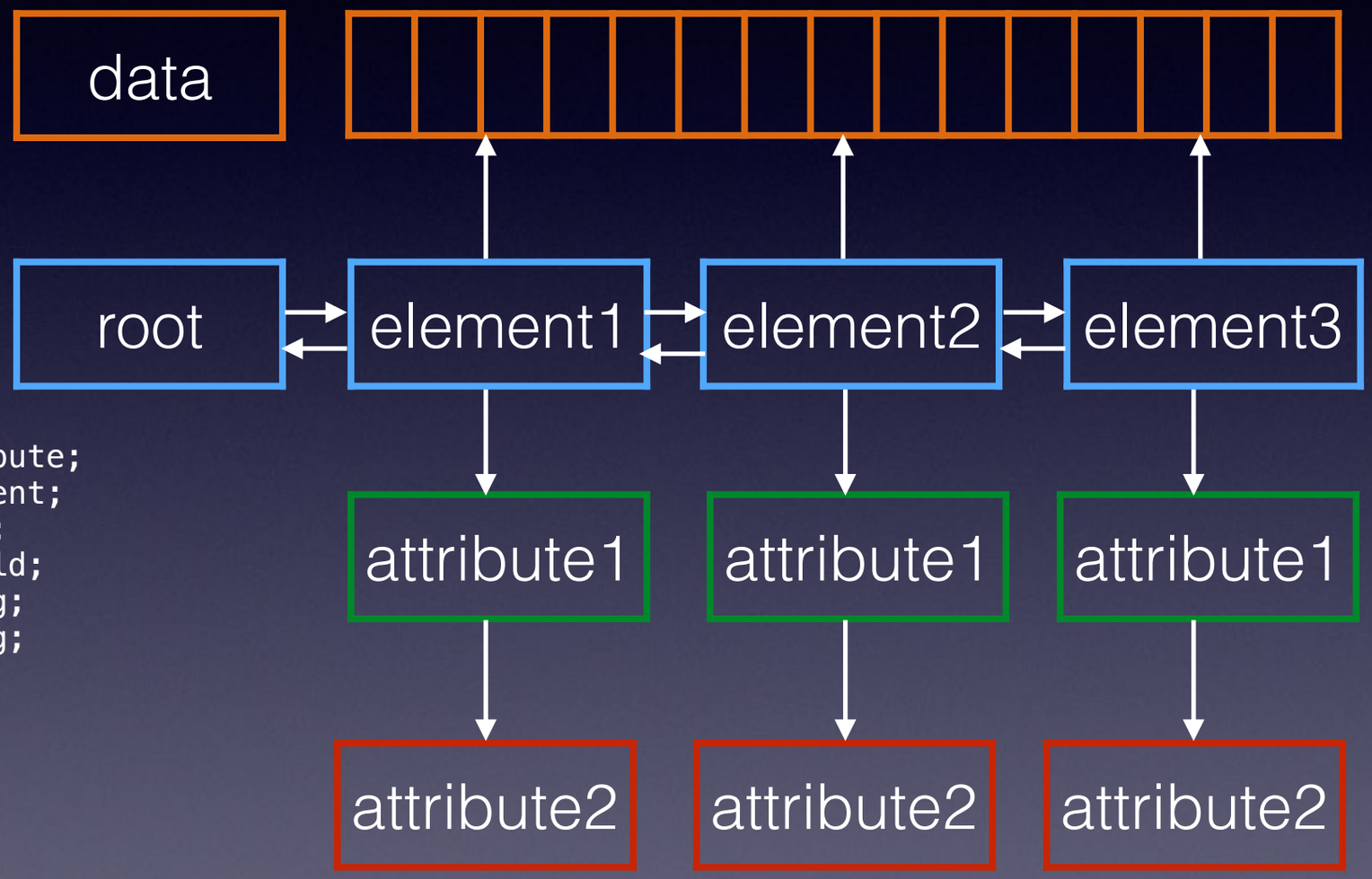
- BISkinType

```
[BISkinType initWithDictionary:(NSDictionary *)dict]

[NSDictionary xxxForKey:(NSString *)key]
```

皮肤编/解码

- BISkinIniParser

```
typedef struct _BISkinAttribute {
    char   *name;
    char   *value;
    struct _BISkinAttribute *next;
} BISkinAttribute;

typedef struct _BISkinElement {
    char   *name;
    struct _BISkinAttribute *firstAttribute;
    struct _BISkinElement   *parentElement;
    struct _BISkinElement   *firstChild;
    struct _BISkinElement   *currentChild;
    struct _BISkinElement   *nextSibling;
    struct _BISkinElement   *prevSibling;
} BISkinElement;
```



- BISkinType

```
[BISkinType initWithElement:(BISkinElement *)element]

[BISkinIniParser attributeIntegerValue:(BISkinAttribute)attribute];
```

皮肤编/解码

- UIImage *image = +[UIImage imageWithContentOfFile:];
- UIImageView.image = image;
- UIKit 创建implicit CATransaction
- 主线程RunLoop下次结束时CoreAnimation提交该transaction.
  - 分配内存Buffer
  - 图片数据从文件读入到内存Buffer
  - 解码图片数据
  - 渲染解压完的图片数据

皮肤编/解码

- 创建图片

[UIImage imageNamed:]

```
UIImage *image = [UIImage imageWithContentsOfFile:filePath];
CGImageRef imageRef = CGImageCreateWithImageInRect([image CGImage], rect);
UIImage *ret = [UIImage imageWithCGImage:imageRef];
CGImageRelease(imageRef);
```

- 保存图片数据

```
CGImageRef imageRef = [image CGImage];
CGDataProviderRef dataProvider = CGImageGetDataProvider(imageRef);
CFDataRef dataRef = CGDataProviderCopyData(dataProvider);
const UInt8 *buffer = CFDataGetBytePtr(dataRef);
fwrite(buffer, 1, count, fp);
```

皮肤编/解码

问题1: 切片性能差

```
[UIImage imageWithContentsOfFile:filePath];            Compressed   Data
CGImageGetDataProvider() & CGDataProviderCopyData();   Uncompressed Data
```

Problem : iOS maintains the uncompressed data

`CGImageCreateWithImageInRect`

解决：强制解码

```
CGContextRef context = CGBitmapContextCreate(NULL,width,height…);
CGContextDrawImage(context, rect, image.CGImage);
CGImageRef imageRef = CGBitmapContextCreateImage(context);
UIImage   *imageOut = [UIImage imageWithCGImage:scale:orientation:];
```

皮肤编/解码

问题2: 内存效率低

- 内存消耗大
- 内存频繁Create & Destroy，性能差

解决 - 使用自定义的Buffer or CGBitmapContextGetData()

```
CGContextRef context = CGBitmapContextCreate(buffer,width,height…);
CGContextDrawImage(context, rect, image.CGImage);

char *src = buffer + start;
for (int i = 0; i < height; i++) {
    memcpy(dst, src, length);
    dst += length;
    src += bytesPerRow;
}
```

从图片数据中得到切片数据，并保存到文件中，不需要生成UIImage对象

面板重用

- 将数据层和视图层分离
- 数据层使用单例(为了节省内存及数据一致性)
- 每个输入App都会产生一个视图层
- panel对象由对象池中获取

面板重用

BICore

ImageSou

DataMgr

BIToolBar

BIPanelView

BICoreMgr

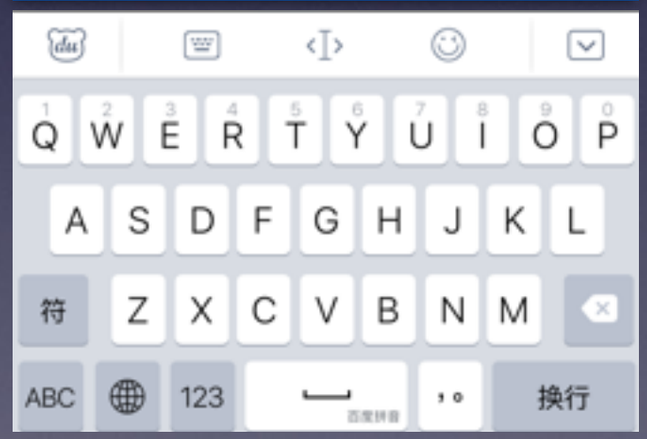
RootController

BIToolBar

BIPanelView

BICoreMgr

RootController

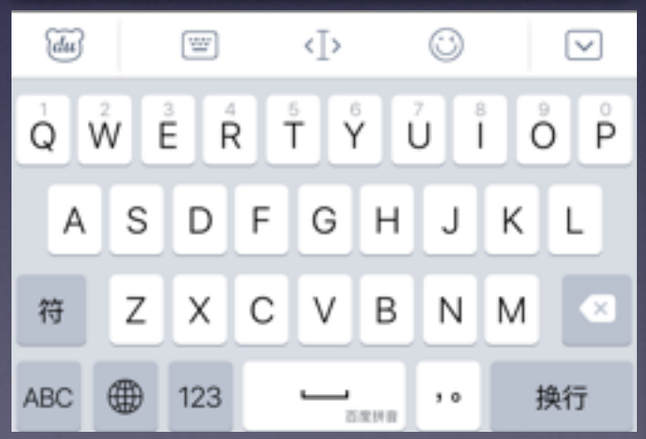


BIToolBar

BIPanelView

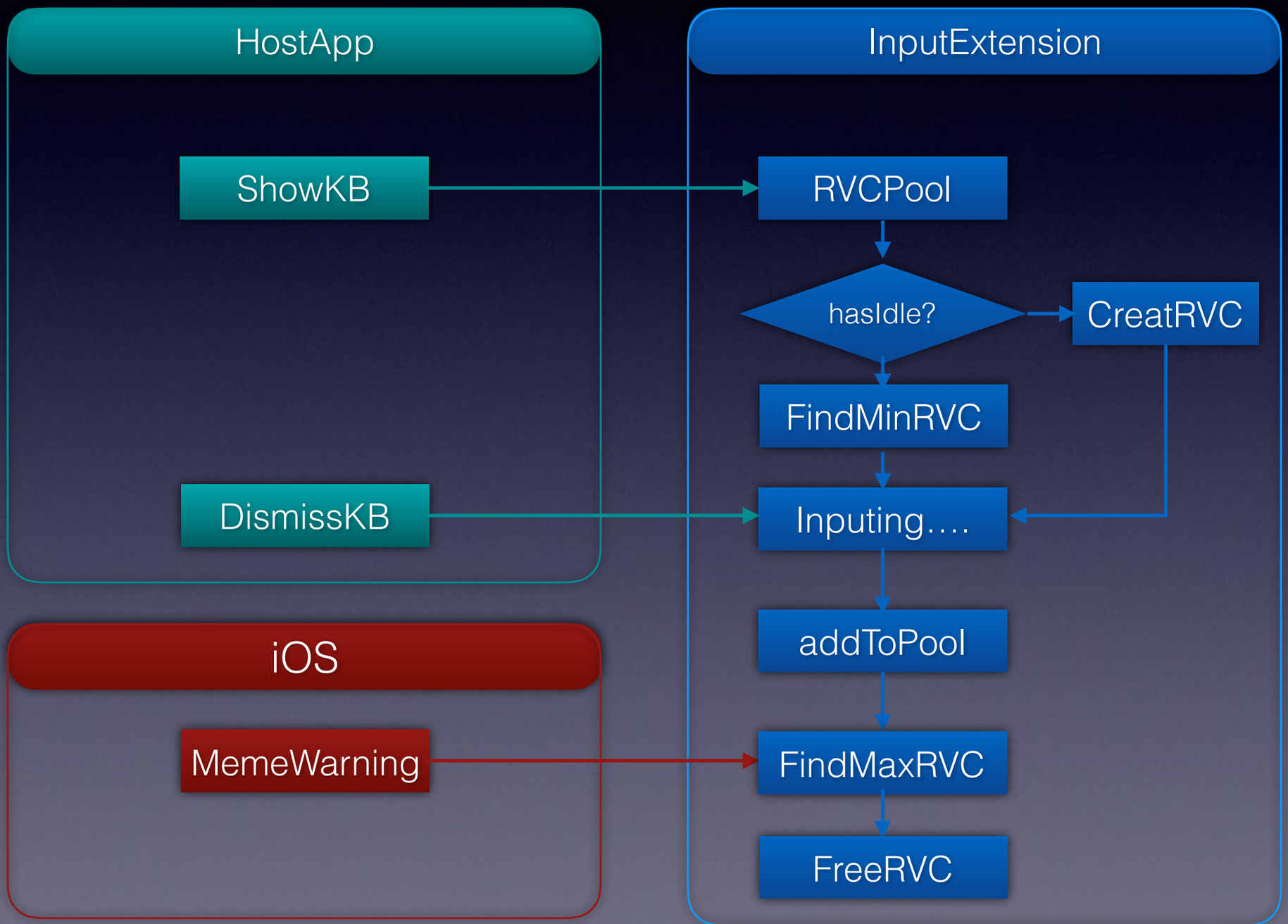
BICoreMgr

RootController



Baidu 百度

面板重用


HostApp
ShowKB
DismissKB
iOS
MemeWarning
InputExtension
RVCPool
hasIdle?
CreatRVC
FindMinRVC
Inputing….
addToPool
FindMaxRVC
FreeRVC

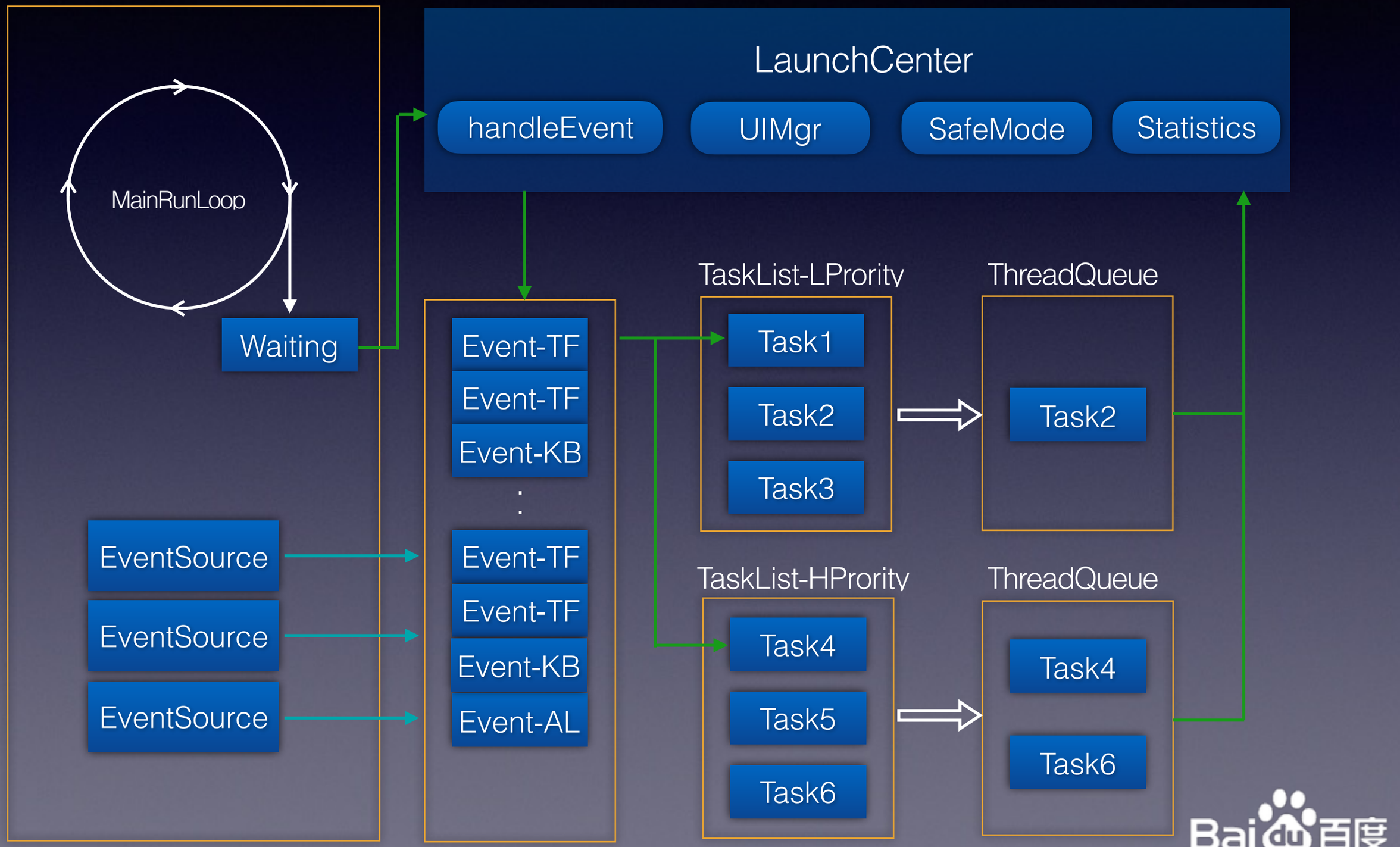
次任务调度

- 为了不影响输入,静默任务被添加到面板启动过程中
- 严重影响了面板的启动时间和稳定性
- 会出现面板上多次弹窗等问题

次任务调度

- 解决方案:
  - 将每个功能抽象成一个任务
    - 每个任务配置自己的触发条件,监控事件,所需资源,优先级等信息
- 调度模块会选择合理的时间调度任务
- 调度模块会调整任务优先级,统计运行时间
- 当任务崩溃超过一定次数时,下次启动就会卸载该任务

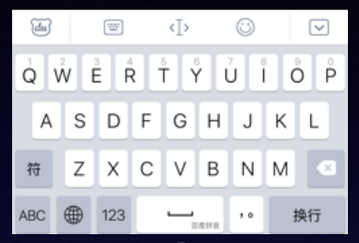
次任务调度


LaunchCenter
handleEvent
UIMgr
SafeMode
Statistics
MainRunLoop
Waiting
Event-TF
Event-TF
Event-KB
Event-TF
Event-TF
Event-KB
Event-AL
EventSource
EventSource
EventSource
TaskList-LPrority
Task1
Task2
Task3
ThreadQueue
Task2
TaskList-HPrority
Task4
Task5
Task6
ThreadQueue
Task4
Task6

内核交互优化

- 交互反馈实时性要求高
- 耗时操作
    - 初始化、词库加载安装等
    - 键盘类型的切换
    - 候选字查询
- 单内核多键盘
    - 支持多session
    - 内存的限制

内核交互优化



showKB | loadSkin | … | keyTap | switchKB | dicUpdate | keyTap | Main RunLoop

low priority jobs cancel

init | loadDic | extra jobs | query | reload | installDic | query | Core Queue

NextKBShow

Async | Sync

BICore

Thanks