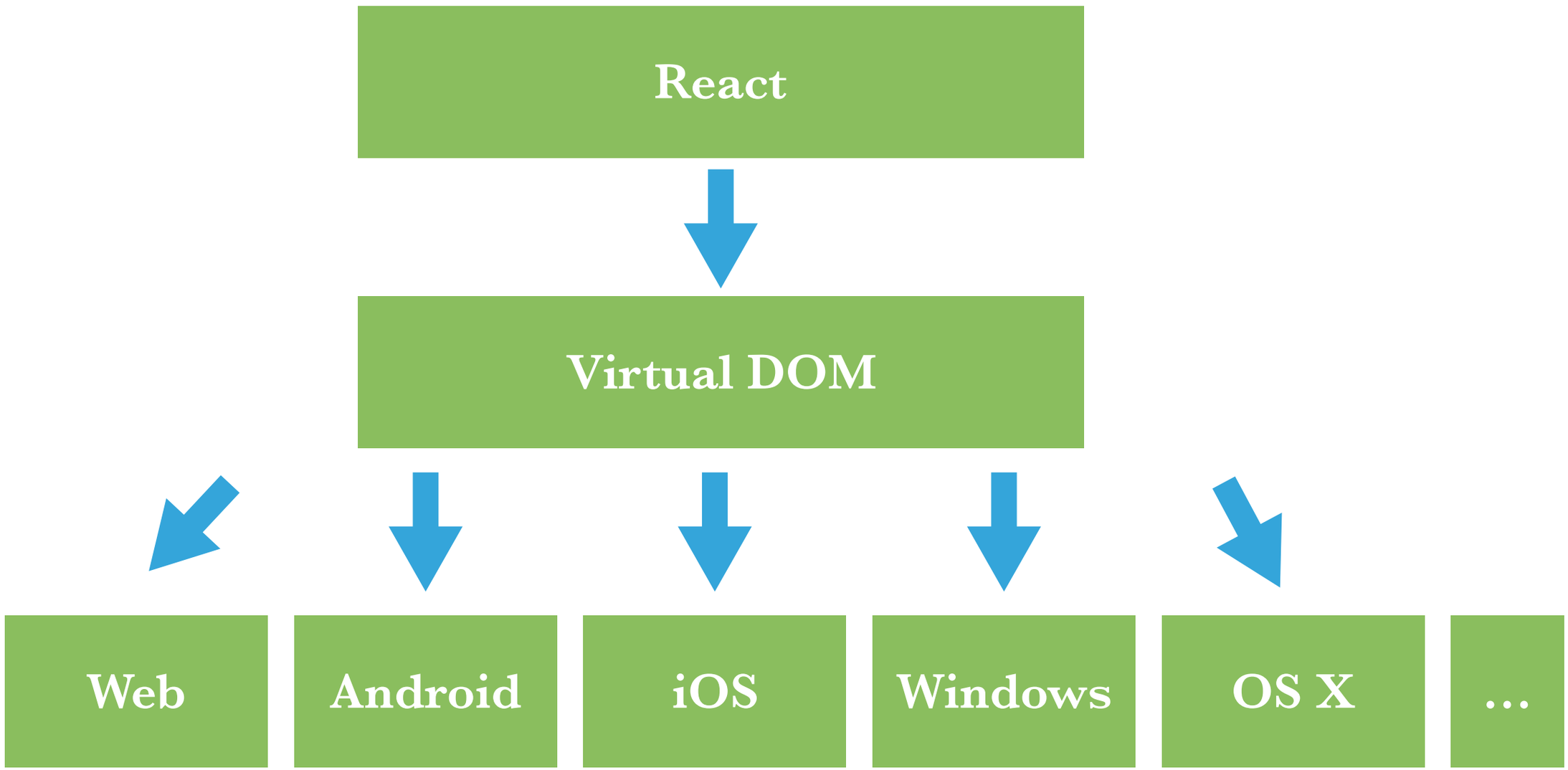
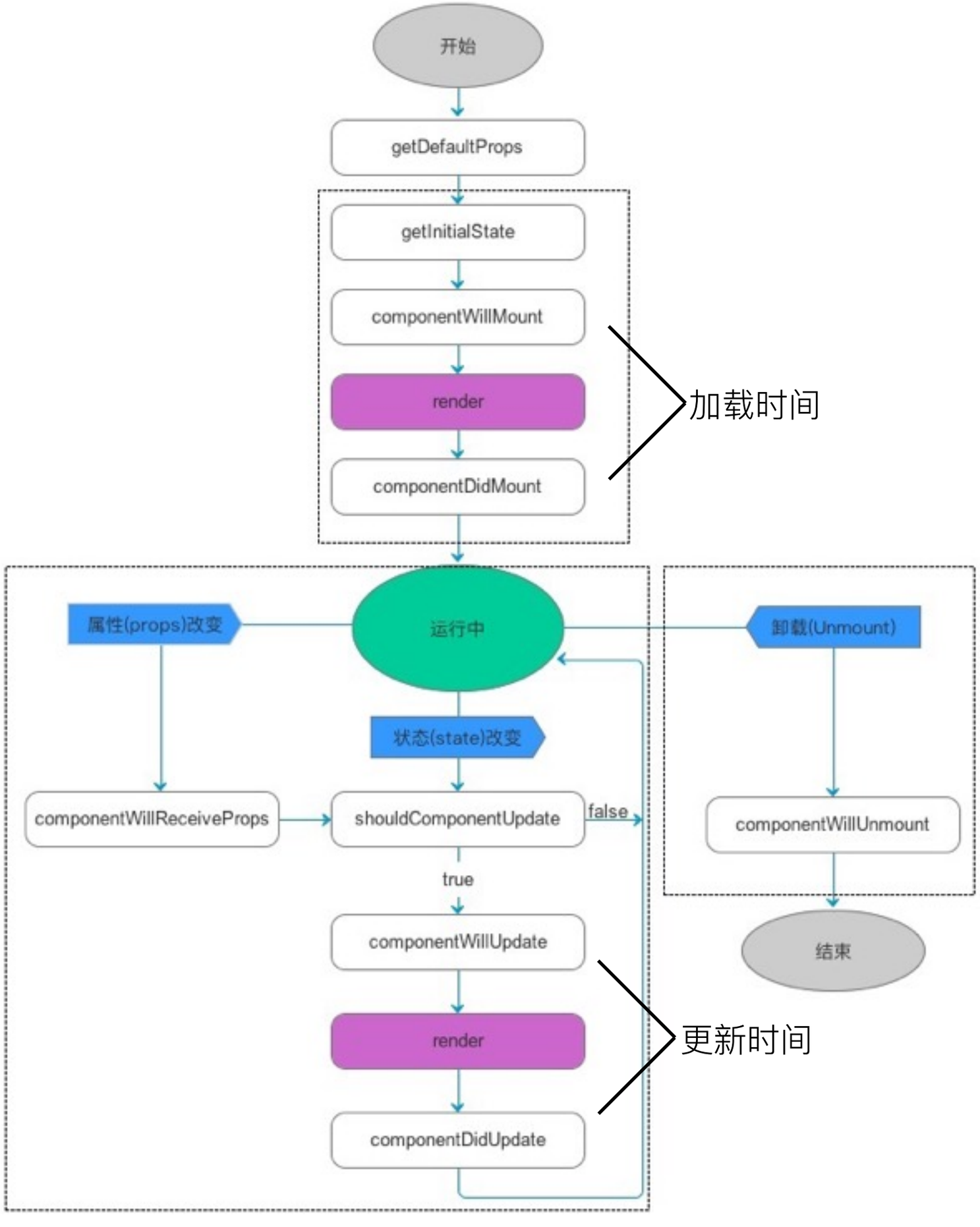
RN 性能分析实践

吴晶 @Android笔记

目录

概述




开始
getDefaultProps
getInitialState
componentWillMount
render
componentDidMount
加载时间
属性(props)改变
运行中
卸载(Unmount)
状态(state)改变
componentWillReceiveProps
shouldComponentUpdate
false
componentWillUnmount
true
componentWillUpdate
render
componentDidUpdate
结束
更新时间

概述

- 性能评价
  - 加载时间
  - 更新时间
  - 消耗内存
  - CPU 负载
  - 绘制延迟*

性能评价

- ScrollView
  - 容器组件
  - 可以交互
  - 支持两个方向
  - 支持 Refresh
  - ListView 的基础

```
import React, {
  ScrollView,
} from 'react-native';
```

```
<ScrollView>
  <Child1 />
  <Child2 />
  <Child3 />
  ...
</ScrollView>
```

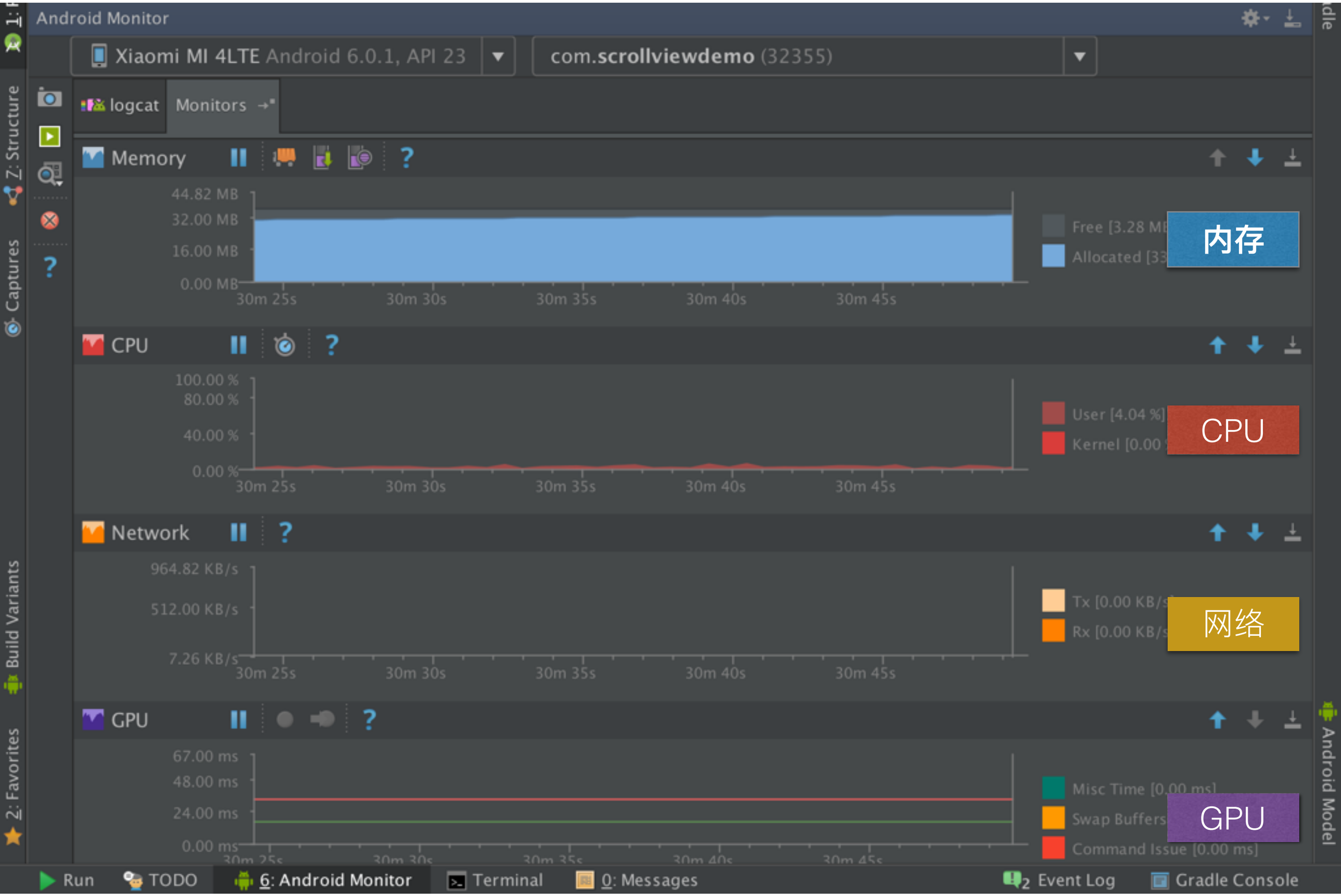
性能评价

```
const N = 5;
class ScrollViewTest extends Component {
  startTime: Date;
  updateTime: Date;

  componentWillMount() {
    this.startTime = new Date();
  }
  componentDidMount() {
    log('load time: ' + (new Date().getTime() - this.startTime.getTime()));
  }
  componentWillUpdate(nextProps: any, nextState: any, nextContext: any) {
    this.updateTime = new Date();
  }
  componentDidUpdate(prevProps: any, prevState: any) {
    log('update time: ' + (new Date().getTime() - this.updateTime.getTime()));
  }
  render() {
    let children = [];
    for (var i = 0; i < N; i++) {
      children.push(<View tag={"T" + i} key={"key_" + i} style={styles.child} />)
    }

    return (
        <ScrollView style={styles.scrollView}>
          {children}
        </ScrollView>
    );
  }
}
```

性能评价

- **Logcat output:**

```
05-27 21:04:32.530 32355-9781/com.scrollviewdemo I/ReactNativeJS: [ScrollViewTest] load time: 850
…
05-27 21:04:45.529 32355-9781/com.scrollviewdemo I/ReactNativeJS: [ScrollViewTest] update time: 130
```


Android Monitor
Xiaomi MI 4LTE Android 6.0.1, API 23
com.scrollviewdemo (32355)
logcat
Monitors
Memory
44.82 MB
32.00 MB
16.00 MB
0.00 MB
30m 25s
30m 30s
30m 35s
30m 40s
30m 45s
Free [3.28 MB
Allocated [33
内存
CPU
100.00 %
80.00 %
40.00 %
0.00 %
30m 25s
30m 30s
30m 35s
30m 40s
30m 45s
User [4.04 %]
Kernel [0.00
CPU
Network
964.82 KB/s
512.00 KB/s
7.26 KB/s
30m 25s
30m 30s
30m 35s
30m 40s
30m 45s
Tx [0.00 KB/s
Rx [0.00 KB/s
网络
GPU
67.00 ms
48.00 ms
24.00 ms
0.00 ms
Misc Time [0.00 ms]
Swap Buffers
Command Issue [0.00 ms]
GPU
Run
TODO
6: Android Monitor
Terminal
0: Messages
2 Event Log
Gradle Console
1: Project
Z: Structure
Captures
Build Variants
2: Favorites
Android Model

Android Device Monitor

Quick Access

DDMS | Hierarchy View

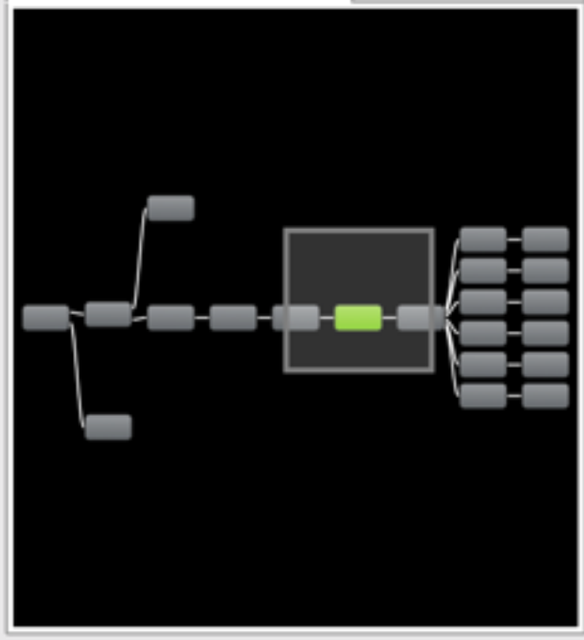
Windo... | View ...

Property	Value
Accessibility	
Drawing	
Events	
Focus	
Layout	
Measurement	
Miscellaneous	
Padding	
Scrolling	
Text	
Theme	

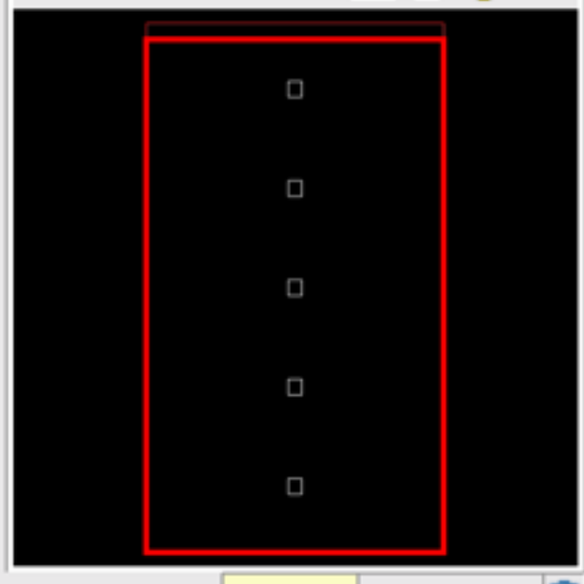
Tree View

Filter by class or id: | 20% | 200%

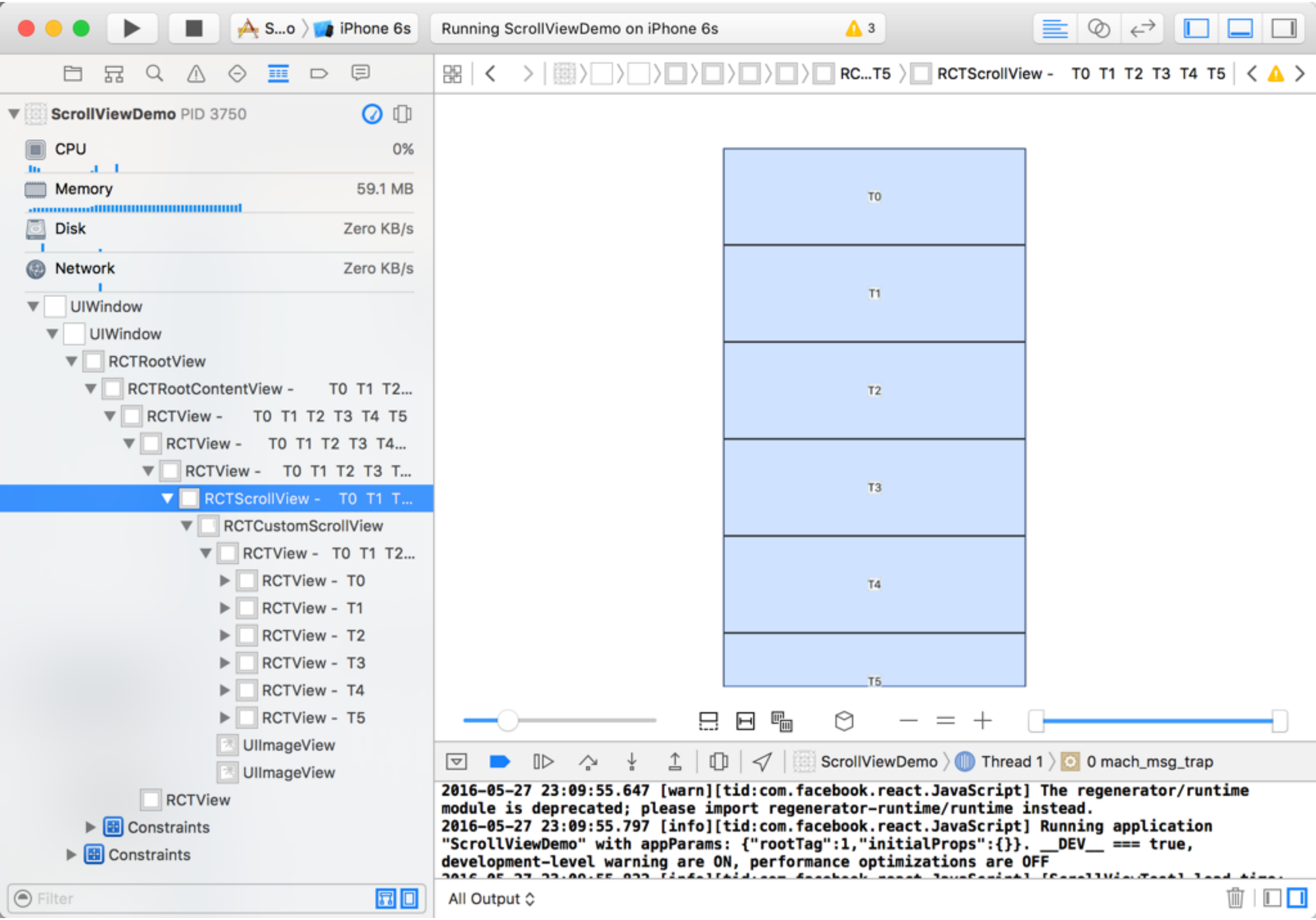
Tree Overview



Layo... | Cons...



261M of 620M


S...o
iPhone 6s
Running ScrollViewDemo on iPhone 6s
3
RC...T5
RCTScrollView - T0 T1 T2 T3 T4 T5
ScrollViewDemo PID 3750
CPU 0%
Memory 59.1 MB
Disk Zero KB/s
Network Zero KB/s
UIWindow
UIWindow
RCTRootView
RCTRootContentView - T0 T1 T2...
RCTView - T0 T1 T2 T3 T4 T5
RCTView - T0 T1 T2 T3 T4...
RCTView - T0 T1 T2 T3 T...
RCTScrollView - T0 T1 T...
RCTCustomScrollView
RCTView - T0 T1 T2...
RCTView - T0
RCTView - T1
RCTView - T2
RCTView - T3
RCTView - T4
RCTView - T5
UIImageView
UIImageView
RCTView
Constraints
Constraints
Filter
T0
T1
T2
T3
T4
T5
ScrollViewDemo
Thread 1
0 mach_msg_trap
2016-05-27 23:09:55.647 [warn][tid:com.facebook.react.JavaScript] The regenerator/runtime module is deprecated; please import regenerator-runtime/runtime instead.
2016-05-27 23:09:55.797 [info][tid:com.facebook.react.JavaScript] Running application "ScrollViewDemo" with appParams: {"rootTag":1,"initialProps":{}}. __DEV__ === true, development-level warning are ON, performance optimizations are OFF
All Output

性能分析

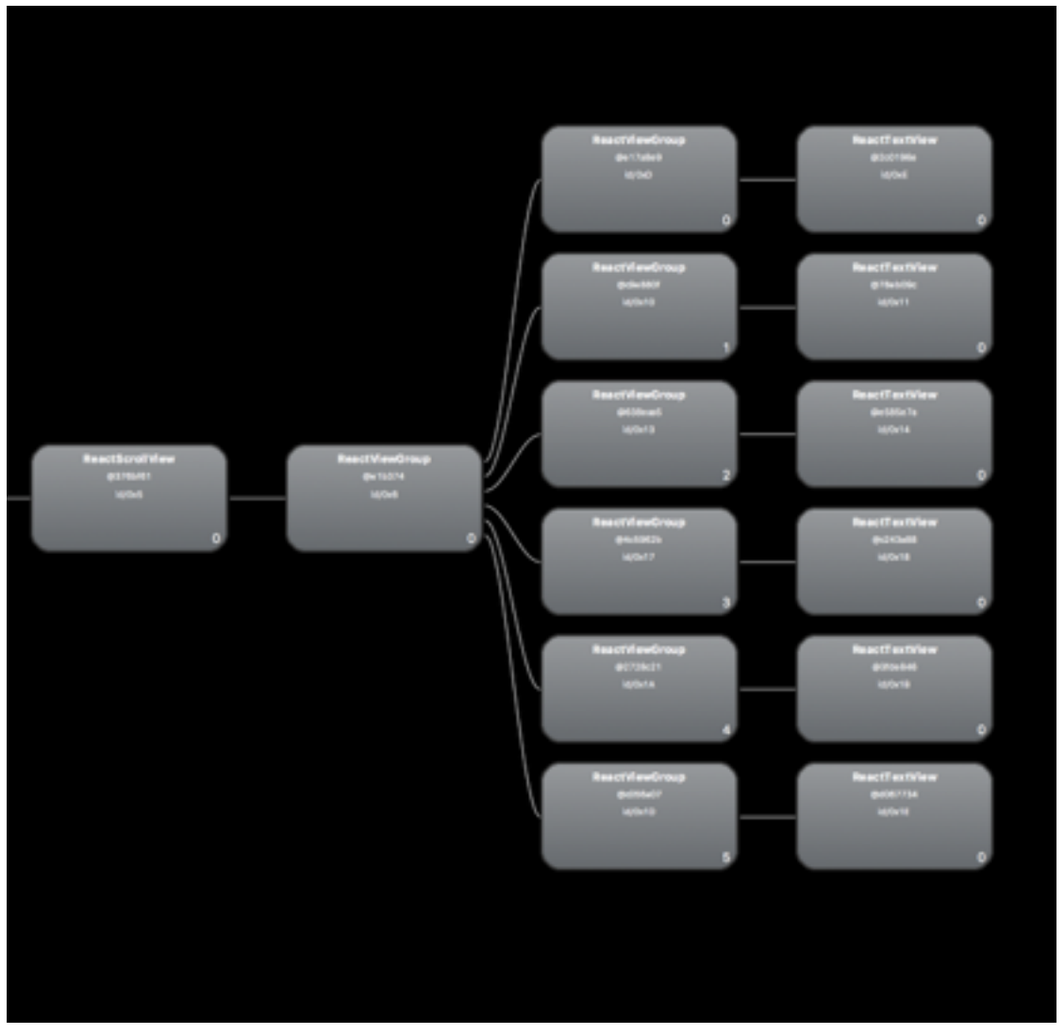
- N = 10，100，1000

```
render() {
  let children = [];
  for (var i = 0; i < N; i++)
    children.push(<View tag={"T" + i} key={"key_" + i}/>);
  return (
      <ScrollView style={styles.scrollView} >
        {children}
      </ScrollView>);
}
```

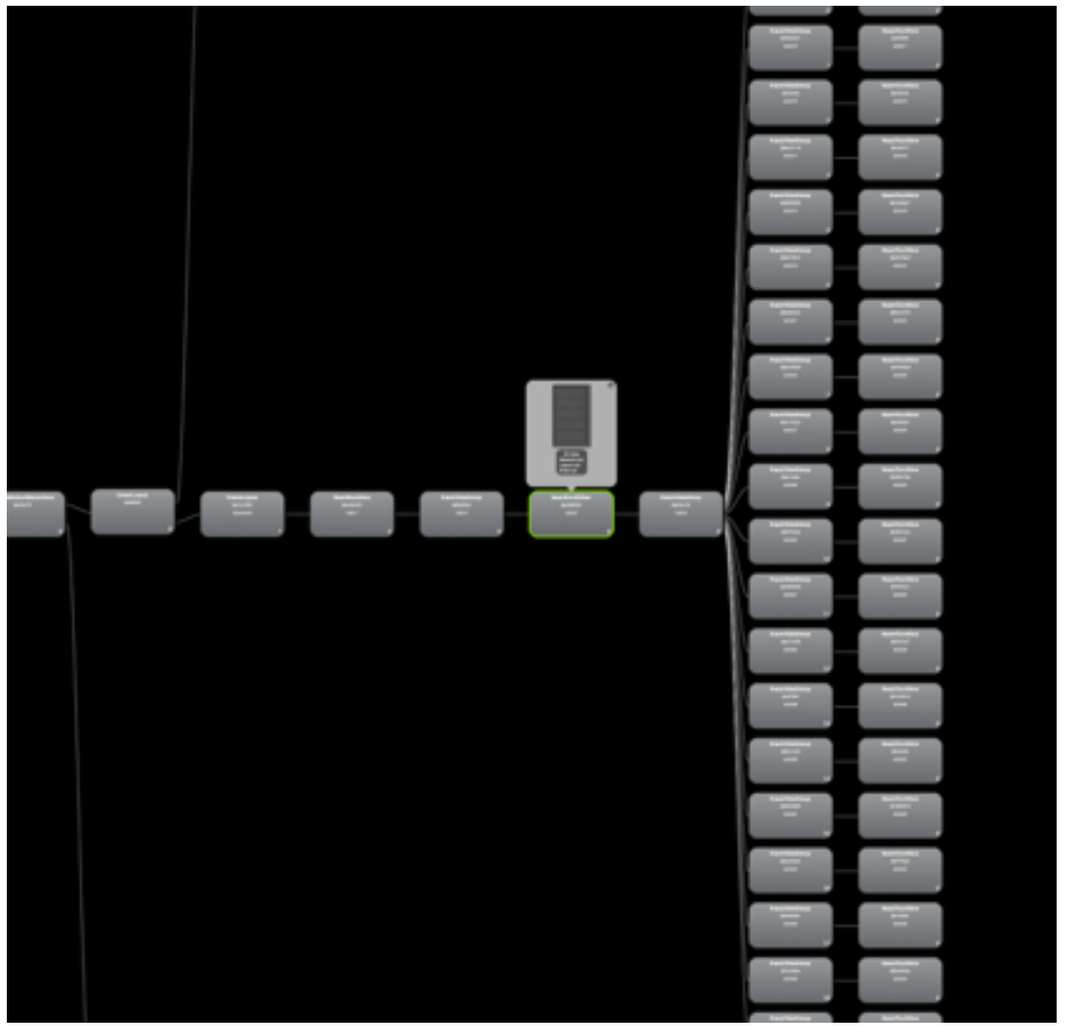
N	加载时间(ms)	占用内存(MB)	绘制时间*(ms)
10	309	19.7	14.666
100	1170	21.9	15.016
1000	9461	26.5	15.025

性能分析

- removeClippedSubviews (true/false)



true



false

性能分析

- 现象
  - 绘制时间基本不变
  - 加载时间随着子 View 的数量递增
- 矛盾
  - 加载时间递增
  - 自动移除溢出的子 View

性能分析

- JS 端源码

```
// ScrollView.js
var AndroidScrollView = …;
var AndroidHorizontalScrollView = …;

var ScrollView = React.createClass({
  render: function() {
    var contentContainer =
       <View ...
         removeClippedSubviews={this.props.removeClippedSubviews}
         collapsable={false}>
         {this.props.children}
       </View>;

     return (
      <ScrollViewClass ...>
        {contentContainer}
      </ScrollViewClass>
    );
  }
});
```

性能分析

- Native 端源码（Android）
  - RCTScrollView 和 AndroidHorizontalScrollView
    - RCTScrollView 继承 ScrollView
    - RCTScrollView 实现 ReactClippingViewGroup 接口

性能分析

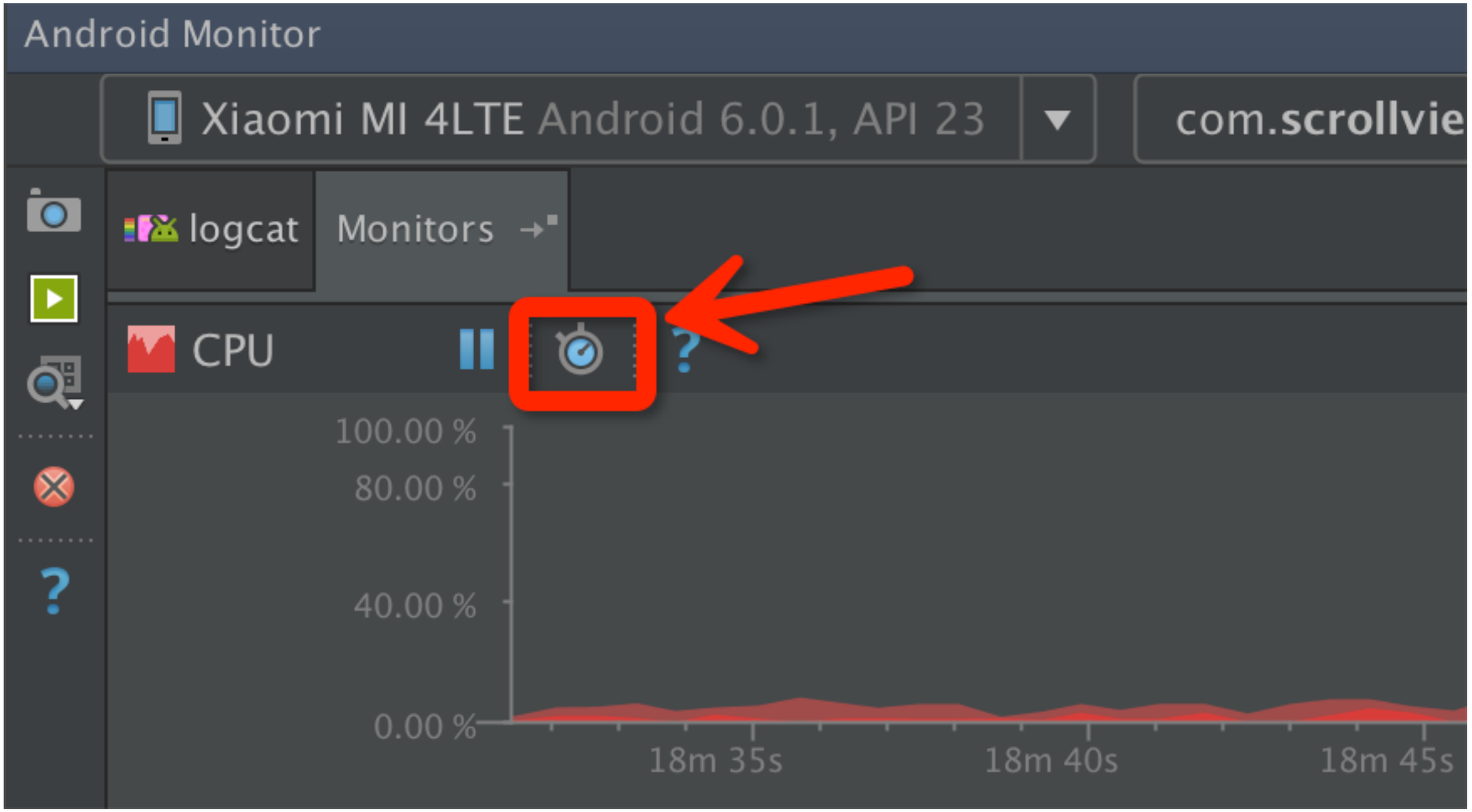
```
@Override
public void updateClippingRect() {
  if (!mRemoveClippedSubviews) {
    return;
  }
  ...
  View contentView = getChildAt(0);
  if (contentView instanceof ReactClippingViewGroup) {
    ((ReactClippingViewGroup) contentView).updateClippingRect();
  }
}
```

```
private void updateSubviewClipStatus(Rect clippingRect, int idx, int clippedSoFar) {
  View child = Assertions.assertNotNull(mAllChildren)[idx];
  sHelperRect.set(child.getLeft(), child.getTop(),
                  child.getRight(), child.getBottom());
  boolean intersects = clippingRect
      .intersects(sHelperRect.left, sHelperRect.top,
                  sHelperRect.right, sHelperRect.bottom);

  if (!intersects && child.getParent() != null && !isAnimating) {
    super.removeViewsInLayout(idx - clippedSoFar, 1);
  } else if (intersects && child.getParent() == null) {
    super.addViewInLayout(child, idx - clippedSoFar, sDefaultLayoutParam, true);
  }
  …
}
```

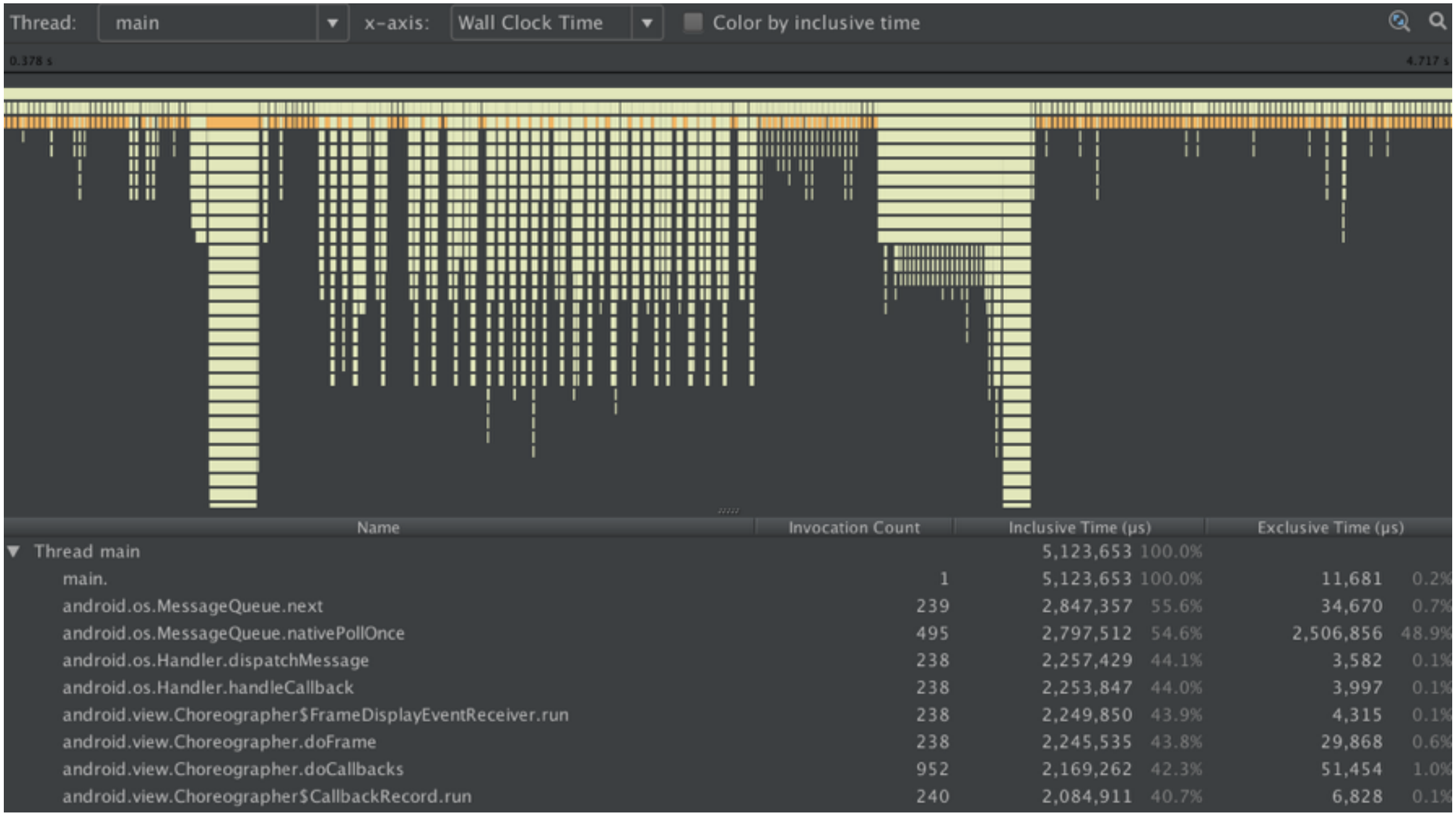
性能分析

```
public class ReactViewGroup extends ViewGroup implements
    ReactClippingViewGroup {
    private boolean mRemoveClippedSubviews = false;
    // 用来保存所有子 View 的数组，包括可见和不可见的
    private @Nullable View[] mAllChildren = null;
    private int mAllChildrenCount;

    // 当前 ReactViewGroup 于父 View 相交矩阵，
    // 也就是它自己在父 View 中可见区域
    private @Nullable Rect mClippingRect;
    ...
}
```

性能分析


Android Monitor
Xiaomi MI 4LTE Android 6.0.1, API 23
com.scrollvie
logcat
Monitors
CPU
100.00 %
80.00 %
40.00 %
0.00 %
18m 35s
18m 40s
18m 45s

性能分析



Name	Invocation Count	Inclusive Time (µs)		Exclusive Time (µs)	
▼ Thread main		5,123,653	100.0%		
main.	1	5,123,653	100.0%	11,681	0.2%
android.os.MessageQueue.next	239	2,847,357	55.6%	34,670	0.7%
android.os.MessageQueue.nativePollOnce	495	2,797,512	54.6%	2,506,856	48.9%
android.os.Handler.dispatchMessage	238	2,257,429	44.1%	3,582	0.1%
android.os.Handler.handleCallback	238	2,253,847	44.0%	3,997	0.1%
android.view.Choreographer$FrameDisplayEventReceiver.run	238	2,249,850	43.9%	4,315	0.1%
android.view.Choreographer.doFrame	238	2,245,535	43.8%	29,868	0.6%
android.view.Choreographer.doCallbacks	952	2,169,262	42.3%	51,454	1.0%
android.view.Choreographer$CallbackRecord.run	240	2,084,911	40.7%	6,828	0.1%

性能分析

- 总结
  - 所有的子 View 都要创建
  - 耗时操作
    - 构建 View
    - Measure
    - Layout/Draw*

性能分析

- ListView
  - 基于 ScrollView 实现
  - 基于 RecyclerView 实现
- 不同点
  - 减少首次加载个数
  - 逐渐添加

性能分析

- ListView 性能
  - 加载速度比 ScrollView 明显快
  - 滑动列表会有延迟
  - 随着向下滑动，不断创建子 View
  - 长列表，内存可能会消耗完
  - 被社区诟病

性能分析

- ViewPagerAndroid
  - 类似 ScrollView
  - 构建所有的页
- 性能问题
  - 加载慢
  - 内存消耗大

性能分析

- React Native 模型
  - 虚拟 Dom 和 Native 组件“几乎”一一对应
  - 不够智能，不能按需构建
- 导致的性能问题
  - 加载时间长
  - 内存消耗大

优化

- 思路
  - Native 的 ListView/RecyclerView
  - 按需构建子 View
  - 重复利用 View 对象

优化

- 按需加载
  - 在业务实现上，按需添加子项目
- 节省内存
  - 尽量简化不可见的 View
  - https://github.com/sghiassy/react-native-sglistview

优化

- 终极解决方案
  - 智能的虚拟 Dom
    - 按需创建
- 智能的 Native 组件
  - 重复利用

总结

- RN 性能
  - 比 H5 好很多
  - 性能问题确实存在
  - 存在优化的可能性
  - Virtual DOM

谢谢

微博 @Android笔记