Optimization of Mobile Application with Instruments

Priyanka Kawle¹, Prof. Mona Mulchandani²
¹ PG, Dept. of CSE, Jhulelal Institute of Technology, NAGPUR
Email: priyanka911.kawle@gmail.com
² Asst. Prof., Dept. of CSE, Jhulelal Institute of Technology, NAGPUR
Email: mona_mulchandani@yahoo.co.in

Abstract - In today's world everyone wants their application to work smoothly and it should be appreciated as well. It's not an easy task for the developers to achieve it. In this competitive market the performance of the application matters a lot. The interface of the application should be flawless, no lags when any animation is done, app should be fast and responsive. So in this paper we will measured our application in areas like CPU, GPU, Memory, Performance and Tim Profiling.

1. Introduction
Apple has provided us with a tool called as ‘Intruments’. It comes with the xcode tool set and its very powerful and flexible testing tool. It's a tool which understand your device and application to optimize their performance and behaviour. One should include this tool in the early stage of the development cycle to find issues and save time. Detail analysis results are provided by collecting application and device data which we can go through side by side. Lets look at the Instrument in details.

1.1 Instruments
After discussing about instrument now lets discuss how we have utilize this tool in our application for optimization. Below Fig.1 shows how to open an intrument from IDE i.e Xcode which is used for developement of iOS app.

![Fig. 1 Instruments Launching](image)

Once we launched the instrument tool, Fig.2 shows several options on which the application needs to be tested. It totally depends on which area the application needs to be tested. This options includes File activity, Cocoa layout, Core data, Energy log, Network, Memory leaks, Counters and may more in the list.

![Fig. 2 Instruments Options](image)
1.2 CPU Usage
The first option is the CPU usage. If we look at the below Fig.3 we can see several sections. First, the graphics presentation were spikes are displayed which indicates the CPU utilization. The second part is the call trace, again the stack trace is divided in 3 parts. The left pane displays the duration, middle one displays methods name and the right pane indicates the events. If we see in the left pane most CPU utilization is during application start (263.00 ms), main function(163.00ms) which is normal for application. The below durations are for asynchronous loop which are used for animations in the app. Most of the times its on higher side but Swift works at core level which reduces the overheading.

1.3 GPU Usage (Graphic Activity)
The Graphic usage represent the view rendering duration. The blue horizontal row displays the rendering of view frames per second. The application views are created in Story board and contrainst are applied properly to rendered in exact view reducing GPU duration.

1.4 Memory Usage
This activity is very important to check and monitor memory allocation. The above graphic shows the complete heap allocation of the application and below it is the leak section which is nil in our application. Zero leaks is the ultimate target of the developer while we have achieved.

1.5 Performance
The over all application performance is very important. This involves the user activity, system activity, virtual memory activity and the overall usage of the CPU activity. The below fig.6 display loading performance of the application which are divided into four sections mentioned above.
By going through the stack of our application the results are very promising compare to Objective C. The overall performance of our application when comparing with memory usage, CPU utilization are better.

2. Application Architecture

The iOS app are hugely base on the basic architecture provided by apple i.e MVC (Model-View-Controller)

Fig.7 shows you the layout of the MVC pattern

2.1 Model

Model are nothing but the data which is presented or displayed on the view. But for this purpose the Model never interact with the View. If we see the Fig.7 the model interact with the controller inorder to display the data. In our application we have created models for Department, Academy, About info and etc

2.2 Controller

The Controller is the mediator for the View and Model. If any data need to represent on View it will ask Model to provide the data. If any action occured on the View that action is handled by Controller and will perform action depending upon the event.

In the fig.7 the Controller received and update action from the View, now the Controller will notify model to update by providing the data received from the View.

2.3 View

View are responsible for displaying or presenting the layout of the screen. We call this layer as presentation layer. In our application we have used storyboard to interact with other screens.

We have used independent .xib file wherever required for proper use of the data to display. We have added constraint to the screen for rendering on all size of iPhone.

The Fig 8 mentioned below shows the application life cycle and displays the internal function integration and calling event.

3. Conclusion

The developement of project is done in Swift langauge which explore the features Apple had provided. But implementing it in our application has proved us right where we the figures and graphs are quite better compare to Objective C. As per statistic we have achived

- Better memory utilization
- Zero leaks
- System utilization

References