ENCS3390

OPERATING SYSTEMS

Prepared by:

Abdelrhman Abed 1193191

Instructor: **Dr . Abdel Salam Sayyad**

Section: 2

Certainly! Two of the most widely used mobile operating systems are Android and iOS, and they vary in a number of ways. These two operating systems are contrasted and compared in the following ways based on performance, customizability, and business models:

Customizability:

-Android: Android is known for its high level of customizability. Users can personalize their devices by installing custom launchers, widgets, and changing system settings. Users can sideload apps from sources other than the official Google Play Store, which provides greater freedom but also poses security risks. Developers can create and distribute apps more easily, given the open nature of the platform.

-IOS: offers limited customization options compared to Android. Users can only change wallpapers, rearrange app icons, and use widgets within predefined areas .Apple tightly controls app distribution through the App Store, ensuring a more secure and controlled ecosystem. This limited customization contributes to a more consistent and user-friendly experience.

Performance:

-Android :Because Android smartphones come in such a broad variety of hardware and software combinations, performance can vary greatly between them .While lower-end Android smartphones could have trouble with resource-intensive applications, certain high-end models can deliver remarkable performance. With Project Treble and other enhancements, Android has grown over time, but fragmentation is still an issue.

-IOS: Because Apple designs and manufactures iOS devices, hardware and software may be tightly integrated All iOS devices benefit from this constant high performance and smoothness. Apple is able to tune its products for effective performance since they have control over both the hardware and software.

Business Model:

-IOS(Apple): Through a mix of in-app purchases, hardware sales, app sales, and services, Apple generates revenue from iOS. The selling of iPhones, iPads, and other Apple products is the company's main source of income. Additionally, Apple keeps a portion of all program sales and in-app transactions made via the program Store. Apple also makes money from paid membership services, such as Apple TV+, Apple Music, iCloud, and Apple Arcade.

-Android(Google): Google does not directly profit from the sale of the Android operating system; instead, it is an open-source project. With Android, Google primarily makes money via data collecting and advertising. In addition to providing Google services like search and Maps, they collect user data to enhance their ad targeting. Through app sales and in-app advertisements, the Google Play Store also makes money. A portion of these purchases go to Google.

In programming and execution contexts, two distinct compilation techniques are used: Ahead-of-Time (AOT) and Just-in-Time (JIT). Android compilation is mostly done via AOT, but before we get into it, let's talk about how AOT and JIT compilation vary from one another.

AOT Compilation:

Before being executed, code in an AOT compiler is converted from a high-level programming language into machine code or an intermediate form. The compiled code is saved as a binary file and is executed prior to the program being executed. AOT compilation is generally related to bytecode languages (like Java) and traditional compiled languages (like C, C++) (with tools like the Android DEX compiler).

JIT Compilation:

JIT compilation compiles code during runtime, right before it is performed, as opposed to compiling it beforehand. Whenever necessary, the code is converted into machine code or an intermediate form; this created code is then stored for later usage. Languages like JavaScript, Java (and sometimes Java Virtual Machine, or JVM), and.NET are frequently used in JIT compilation.

Why Android Uses AOT Compilation:

1-Performance: Since AOT-compiled code is already in machine code when it is executed, it often runs more quickly. As a result, programs operate more smoothly by lowering the overhead of dynamic compilation and interpretation.

2- Security: Because the code is precompiled and examined before being executed, AOT compilation can assist increase security by identifying and removing possible security flaws early on.

3- Resource Efficiency: JIT compilation can use a lot of CPU and memory resources during runtime, and Android devices frequently have limited resources. Better resource management results from reducing runtime overhead through AOT compilation.

Advantages and Disadvantages of AOT and JIT Compilation:

AOT Compilation:

-Advantages:

Improved runtime performance: Because AOT-compiled code is in machine code, it often runs quicker.

Enhanced security: During the compilation process, vulnerabilities may be found. Consistent resource consumption: AOT eliminates runtime overhead.

-Disadvantages:

Longer app startup times: Compiling all code ahead of time can lead to longer app launch times.

Larger binary sizes: AOT-compiled apps may be larger because they include compiled code.

JIT Compilation:

-Advantages: Quicker development cycles: Modifications can be made without requiring a full recompile of the program. Reduced binary sizes: Because code is built dynamically, applications may have reduced binary sizes. Adaptable to runtime environments: Just-in-time (JIT) code optimization is possible depending on the unique hardware and use patterns of each device.

-Disadvantages: Slower initial execution: An application's first execution may be slowed down because code is compiled dynamically. Runtime overhead: During execution, JIT compilation uses up CPU and memory resources. Security issues: Vulnerabilities might not be discovered until after execution since code is generated at runtime.

Native vs. Cross-Platform Mobile Development:

Native Mobile Code:

Definition: Native mobile development is the process of developing code using tools and languages native to a particular mobile platform (e.g., Swift/Objective-C for iOS, Java/Kotlin for Android).

-Advantages: Maximum performance: Because native apps are tailored for a particular platform, they usually operate more quickly and smoothly. Access to platform-specific features: Platform-specific APIs and capabilities are fully available to developers. Consistent user experience: Native applications provide a recognizable and consistent user interface since they follow platform rules.

-Disadvantages: Expense and development time are increased: It can be costly and time-consuming to write different codebases for every platform. Maintenance complexity: It might be difficult to maintain and update two different codebases. Restricted code reuse: Android and iOS apps can share very little to no code.

Cross-Platform Mobile Code:

-Definition: Developers may create code once and have it execute on several platforms using cross-platform mobile development. Usually, a single codebase built in a cross-platform language or framework is used.

-Advantages: Faster development: By allowing code to be shared between platforms, development costs and time are decreased. Simpler maintenance: By applying updates and bug fixes to the shared codebase, maintenance may be made easier. Enhanced code reuse: A substantial amount of code may be repurposed by developers across several platforms.

-Disadvantages: Possible trade-offs in performance: Cross-platform apps might not be as optimized or quick as native programs. Restricted availability of platform-specific features: Not all platform-specific APIs may be instantly accessible using cross-platform frameworks.

understanding curve: Compared to typical native programming, understanding the cross-platform framework may be more difficult for developers.

Using Flutter to Create Cross-Platform Apps:

A well-liked open-source framework for creating cross-platform mobile applications is called Flutter. It enables programmers to use a single Dart codebase to create apps that run on both iOS and Android devices. This is what Flutter does to do this:

-Widget-based UI: The user interface of Flutter is created using a widget-based methodology and is subsequently transformed to native components on each platform. This guarantees a native interface on iOS and Android.

-Extremely configurable: Flutter offers a wide range of widgets, and developers may alter their appearance and functionality to conform to platform-specific design standards.

-Native performance: Some of the speed trade-offs related to cross-platform development are lessened by the high-performance visuals and fluid animations that Flutter's engine is built to deliver.

-Platform-specific feature access is made possible by Flutter's plugins and packages, which provide developers access to platform-specific APIs and native capabilities as needed.

-Hot Reload: Developers may view changes in real-time with Flutter's hot reload functionality, which makes the development process.

Describe google chrome as an example of multi-threaded application:

One of the best examples of a multi-process, multi-threaded program is Google Chrome, whose architecture is made to offer a stable and responsive surfing environment. Here's how Google Chrome is described as one of these apps:

Multi-Process: Because Google Chrome has a multi-process design, it can execute several separate processes at once. Every open window or tab in the browser runs as a distinct process. This strategy has several benefits:

-Isolation: The stability of the browser as a whole is unaffected if one tab or process crashes or has problems. Because of this separation, a single faulty webpage or extension cannot cause the browser to crash as a whole.

-Security: Chrome improves security by separating processes. It is less probable that malicious code on one webpage will affect open tabs or corrupt the browser as a whole.

-Efficiency: Compared to single-process browsers, operating several processes uses more memory, but in exchange, offers improved responsiveness and speed. Several CPU cores can be used to divide up tasks, which could result in quicker execution.

Multi-Threaded:

To handle different tasks, Chrome uses numerous threads within each of these processes. This multi-threaded method aids in maximizing responsiveness, speed, and resource use. Among Chrome's important threads are the following:

-UI Thread: The UI thread is responsible for managing user interactions, displaying the user interface in the browser, and reacting to user input.

-Renderer Threads: Every tab that is open has a separate renderer thread that is in charge of rendering the tab's content. These threads are in charge of rendering the webpage, running JavaScript, and parsing HTML.

-Network Thread: This thread is in charge of handling network requests and retrieving media files, HTML, CSS, and JavaScript resources.

-GPU Thread: This thread manages rendering operations, such graphics rendering and animation acceleration, that may be sent to the computer's graphics processing unit (GPU).

Advantages of Chrome's Architecture:

-Enhanced Stability: The isolation of each tab makes sure that an issue in one doesn't impact other tabs or cause the browser to crash as a whole.

-Enhanced Security: By enforcing security measures like the Same-Origin Policy more strictly, cross-site scripting threats are prevented from impacting other tabs.

-Improved Performance: By utilizing the capabilities of contemporary multi-core computers, multi-threading and multi-processing provide quicker page loads and more responsive user interfaces.

-Resource Management: Chrome divides work among threads and processes to effectively manage resource allocation.

-Responsive User Experience: Even when one tab is occupied with a resource-intensive operation, the browser may still react to user input because it divides tasks across threads and processes.