**1. Differences between Mobile Computing and Cloud Computing.**

**Cloud computing**,is the ability to run a program or application on many connected computers at the same time.It allows you to store your files and folders in a “cloud” area on the Internet, allowing you access to all of your files and folders wherever you are in the world – but you do need a physical device with Internet access to access it.

**Mobile computing is** being able to use a computing device even when being mobile and therefore changing location. This could be a laptop or a mobile phone or some device which enables you to telework – working wherever you go because of the small size of the device you’re using. Portability is one aspect of mobile computing.

Cloud computingis used mostly in the context of distributed computing. Mobile computing is the ability to communicate to other devices over a network(telephone/internet) while being mobile.Mobile computing is not as recent as cloud computing. Though there could be network latencies and lags in mobile computing,it is a more clearly understood and widely implemented concept. Cloud computing is relatively new and requires more technologies and back-end support to be scalable.

**Mobile apps may use the cloud for both app development as well as hosting.** A number of unique characteristics of hosted apps make the mobile cloud different from regular cloud computing. Mobile apps may be more reliant upon the cloud to provide much of the computing, storage, and communication fault tolerance than regular cloud computing does.

Understanding these differences helps a lot of predicting and heading off a number of problems with the mobile cloud computing, and enables you to deliver predictable, reliable and fault-tolerant mobile app experiences. Here are some of the key differences:

**Total cloud dependency** - When using mobile cloud computing, apps may rely on the cloud for everything, especially when you are trying to develop the same app to run on multiple platforms at the same time using a browser interface. An example of this is an app that runs on Apple iOS, Android, BlackBerry and Microsoft Windows Phone operating systems. Because of the differences between these platforms, developers may rely on the mobile cloud to perform all of the computing and storage to avoid multiple development and maintenance efforts with individual native apps.

**Mobile cloud computing needs to overcome mobile device differences** –The mobile cloud may need to approximate more or less the same end user experience on all platforms to avoid device specific customizations. Unlike traditional cloud usage, the mobile cloud may need to do these adjustments on its side based on what the mobile device being used is.

**Mobile cloud computing needs to allow for disconnected operation** – Mobile devices may go out of range while an application is being served from the mobile cloud. It needs to support disconnected operation as much as possible.

**Mobile cloud computing needs to be communication fault tolerant** – Mobile connections can become weaker and may even disconnect while an application is being used. The cloud may need to be capable of monitoring the connection strength and needs to be fault tolerant of these possible communications disconnects.

**Distance matters in mobile cloud computing** – Mobile applications when using the mobile cloud may be sensitive to network latencies caused by distance from the server much more than regular cloud computing. The mobile end user experience may suffer if these latencies are too long.

**Mobile cloud computing needs to be mindful of limited energy availability on mobiles** – The mobile cloud needs to be mindful of the limited energy availability on mobile devices and may need to perform all functions on the side of the cloud, rather than expect them to be done on the mobile device.

**Expanded testing capabilities** – The mobile cloud needs to have additional testing capabilities that allow testing for poor network latency, unreliable and intermittent communication with the mobile device, disconnected operation and subsequent synchronization of data with the app on the mobile device.

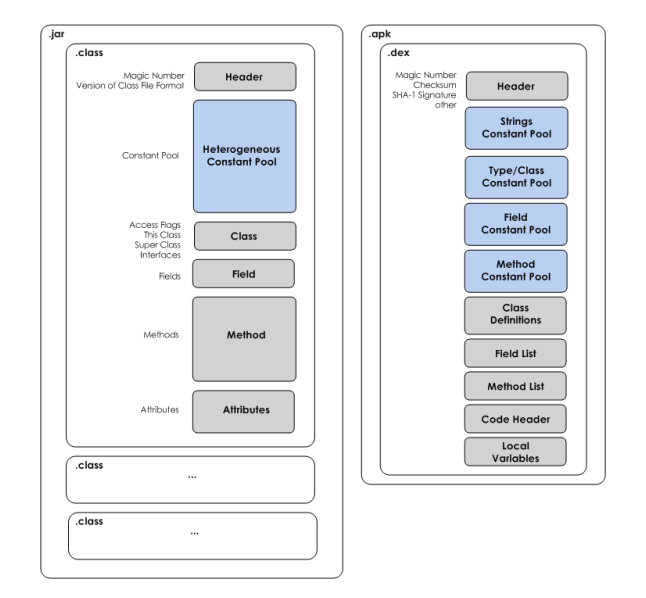
**2. DVM Instructions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode (hex)** | **Opcode name** | **Explanation** | **Example** |
| 01 | Move vx,vy | Moves the content of vy into vx. Both registers must be in the first 256 register range. | 0110 - move v0, v1 Moves v1 into v0. |
| 0F | return vx | Return with vx return value | 0F00 - return v0 Returns with return value in v0. |
| 15 | const/high16 v0, lit16 | Puts the 16 bit constant into the topmost bits of the register. Used to initialize float values. | 1500 2041 - const/high16 v0, #float 10.0 // #41200000 Moves the floating literal of 10.0 into v0. The 16 bit literal in the instruction carries the top 16 bits of the floating point number. |
| 1D | Monitor-enter vx | Obtains the monitor of the object referenced by vx. | 1D03 - monitor-enter v3 Obtains the monitor of the object referenced by v3. |
| 8D | int-to-byte vx,vy | Converts the int value in vy to a byte value and stores it in vx. | 8D00 - int-to-byte v0, v0 Converts the integer in v0 into a byte and puts the byte value into v0. |

1. Vx values in the table denote a Dalvik register.
2. Depending on the instruction, 16, 256 or 64k registers can be accessed.
3. Operations on long and double values use two registers, e.g. a double value addressed in the V0 register occupies the V0 and V1 registers.
4. Boolean values are stored as 1 for true and 0 for false. Operations on booleans are translated into integer operations.
5. All the examples are in hig-endian format, e.g. 0F00 0A00 is coded as 0F, 00, 0A, 00 sequence.
6. The offset can be positive or negative and it is calculated from the offset of the starting byte of the instruction. The offset is always interpreted in words (2 bytes per 1 offset value increment/decrement). Negative offset is stored in two's complement format. The current position is the offset of the starting byte of the instruction.
7. Compare operations returrn positive value if the first operand is greater than the second operand, 0 if they are equal and negative value if the first operand is smaller than the second operand.

**3 .dex File Format**

On the Android platform, Java source code is still compiled into .class files. But after .class files are generated, the “dx” tool is used to convert the .class files into a .dex, or Dalvik Executable, file. Whereas a .class file contains only one class, a .dex file contains multiple classes. It is the .dex file that is executed on the Dalvik VM.



1. The .dex file has been optimized for memory usage and the design is primarily driven by sharing of data.
2. The .dex file format uses shared, type-specific constant pools as it’s primary mechanism for conserving memory.
3. A constant pool stores all literal constant values used within the class. This includes values such as string constants used in your code as well as field, variable, class, interface, and method names. Rather than store these values throughout the class, they are always referred to by their index in the constant pool.
4. In the case of the .class file, each class has its own private, heterogeneous constant pool. It is heterogeneous in that all types of constants (field, variable, class, etc.) are mixed together.
5. Contrast this to the .dex file that contains many classes, all of which share the same type-specific constants pools. Duplication of constants across .class files is eliminated in the .dex format.
6. By allowing for classes to share constants pools, repetition of constant values is kept to a minimum. The consequence of the minimal repetition is that there are significantly more logical pointers or references within a .dex file compared to a .class file.

**4. Application simulating an environment of  context aware computing.**

1. An application that records the daily/hourly temperature ,humidity values and gives the appropriate forecast.
2. A file sharing application which can detect the strongest wi-fi signal in the vicinity or the strongest bluetooth connection and be able to transfer files over both the connectivity frameworks based on signal strength.

Both these applications are simulated in an environment of context aware computing as they measure some sort of perceptible quantities in the environment and react according. They work in a stimulus-response environment. Their computing depends upon the external context. They are sensitive to the environment in which they are simulated.

If the humidity is too high,the first application predicts a rainy day and a cold day if the temperature is low.

The second application automatically transfers files over a bluetooth connection to the communicating device if the wi-fi signal strength is too low.