symmetry

Article

Adaptive Job Load Balancing Scheme on Mobile Cloud Computing with Collaborative Architecture

Byoungwook Kim [1], Hwirim Byun [2], Yoon-A Heo [2] and Young-Sik Jeong [2,*]

[1] Creative Informatics & Computing Institute, Korea University, Seoul 02841, Korea; Byoungwook.kim@inc.korea.ac.kr
[2] Department of Multimedia Engineering, Dongguk University, Seoul 04620, Korea; hazzzly@dongguk.edu (H.B.); hyagood@dongguk.edu (Y.-A.H.)
* Correspondence: ysjeong@dongguk.edu; Tel.: +82-2-2260-3374

Academic Editor: Ka Lok Man
Received: 28 January 2017; Accepted: 25 April 2017; Published: 29 April 2017

Abstract: The adaptive mobile resource offloading (AMRO) proposed in this paper is a load balancing scheme for processing large-scale jobs using mobile resources without a cloud server. AMRO is applied in a mobile cloud computing environment based on collaborative architecture. A load balancing scheme with efficient job division and optimized job allocation is needed because the resources for mobile devices will not always be provided consistently in this environment. Therefore, a job load balancing scheme is proposed that considers personal usage patterns and the dynamic resource state of the mobile devices. The delay time for computer job processing is minimized through dynamic job reallocation and adaptive job allocation in the disability state that occurs due to unexpected problems and to excessive job allocations by the mobile devices providing the resources for the mobile cloud computing. In order to validate the proposed load balancing scheme, an adaptive mobile resource management without cloud server (AMRM) protocol was designed and implemented, and the improved processing speed was verified in comparison with the existing offloading method. The improved job processing speed in the mobile cloud environment is demonstrated through job allocation based on AMRM and by taking into consideration the idle resources of the mobile devices. Furthermore, the resource waste of the mobile devices is minimized through adaptive offloading and consideration of both insufficient and idle resources.

Keywords: mobile cloud computing; collaborative architecture; offloading; mobile resource management; dynamic scheduling algorithm

1. Introduction

Mobile cloud computing (MCC) is a technology that uses computing resources outside of the mobile device [1–5]. For the efficient processing of large-scale jobs, the mobile device must send jobs to an external cloud server. Recently, many studies have been conducted on the technology required for offloading processing jobs to external computer resources and for receiving the processed results, thereby overcoming the hardware limitations of the mobile devices [6–10].

Where the cloud computing architecture uses a server with high computing power, offloading has inherent security problems such as transmission delays and data leakage, which can occur during network-dependent computing data transmission. Therefore, most of the existing studies are focused on the creation of boundaries for security or on the use of relays to improve transmission speed [1,6,8,10,11]. Such offloading methods can be applied efficiently if the server has sufficient computing power and only the transmission of computing data is required. However, no offloading technologies exist for processing large-scale jobs in an environment where the connection with the cloud server is not smooth or where the mobile device computing resources are limited. As an alternative approach to solving the problem of disconnection from the cloud server, the provision of computing services designed only for mobile devices should be researched. In other words, a computing service method is needed that receives resources and services for processing large-scale jobs from other, neighboring, mobile devices and considers the resource state of each mobile device [12–16].

In this research, adaptive mobile resource offloading (AMRO) is proposed, which is a load balancing scheme for processing large-scale jobs using only mobile resources with no external cloud server. This would take place in a mobile cloud computing environment and be based on collaborative architecture among the MCC environments [9]; and would consist of service-oriented architecture, agent-client architecture, and collaborative architecture. This environment would require a load balancing scheme with efficient job division and optimized job allocation, because we cannot assume that the resources of mobile devices will always be provided consistently in this environment. Therefore, a job load balancing scheme is proposed that considers the personal usage patterns and dynamic resource state of the mobile devices. The delay time for computer job processes is minimized through the dynamic job reallocation and adaptive job allocation in the disability state that can arise due to unexpected problems and excessive job allocation to the mobile devices providing the resources for mobile cloud computing. To validate the proposed load balancing scheme, the adaptive mobile resource management without cloud server (AMRM) protocol was designed and implemented, and improved processing speed was verified in comparison to the existing offloading method. Improved job processing speed in the mobile cloud environment is demonstrated through job allocation based on AMRM, which considers the idle resources of the mobile devices. Furthermore, the resource waste of mobile devices is minimized through adaptive offloading taking into consideration both the insufficient resources and the idle resources.

2. Related Work

In this section, the basics of offloading in the existing mobile computing environment and the advantages and disadvantages of offloading methods are described.

Balan et al. [11] investigated the offloading of computations through Spectra, which is an integrated dictionary setup service working through remote procedure call (RPC). Spectra refer to a database that contains the current availability of servers, CPU load, and so on. As an RPC system that combines mobile devices and servers through applications, Spectra proposed a new framework applicable to MCC and a concrete offloading method.

Kemp et al. [17] proposed Cukoo which is a system for offloading mobile device applications in a cloud. They used Java virtual machine (JVM) to implement, in Android, a method of creating a personal mini-cloud consisting of notebook computers or local clusters, such as the commercial offloading service provided by Amazon EC2. They proved that through Cukoo they were able to improve performance and battery efficiency.

Chun et al. [18] proposed CloneCloud, which is a system for automatically converting mobile applications to use a cloud. CloneCloud has implemented a framework for offloading through the network some of the workload of applications to a server with sufficient resources.

Clark et al. [19] implemented a virtual machine migration that sends a memory image from a source server to a destination server. To copy the dictionary without interruption, during the live migration the memory page of the virtual machine provides the illusion of smooth migration to protect the code during the offloading of the program, and security is guaranteed through the boundary of the virtual machine. However, this method has the shortcoming of workload, as it requires too much time for the virtual machine migration, making it too heavy to apply to mobile devices.

Kristensen [20] proposed a framework, Scavenger, which uses the mobile code method for offloading through Wi-Fi. A cost-assessment method was implemented on the basis of the speed of a surrogate server through a scheduler, and mobile devices offload to more than one surrogate server using this framework.

Figure 1 shows Niroshinie Fernando's proposed classification for the offloading of mobile cloud computing into five types from the aspect of functional extension according to structure [9].

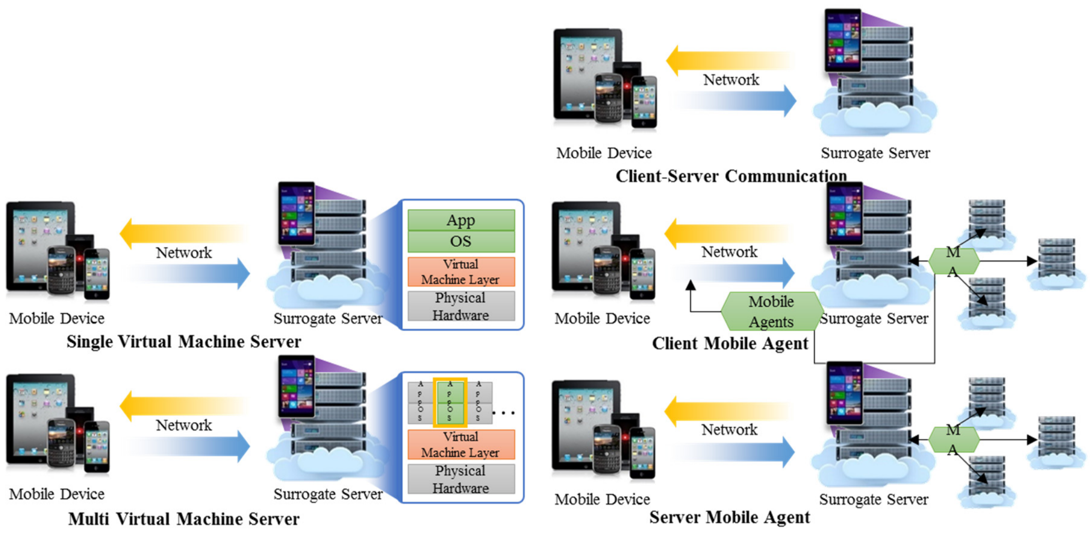


Figure 1. Classification of offloading on mobile cloud computing.

Table 1 outlines the merits and demerits that can be expected when applying these offloading methods to actual MCC [9].

Table 1. Characteristics of offloading methods.

Classification	Merits	Demerits
Client-server communication	Most effective for single service processing	Difficult in coping flexibly with various mobile application environments
Single virtual machine server	Advantageous for single process collaboration	Has issues in processing data such as personal information
Multi virtual machine servers	Advantageous for processing data such as personal information	Inefficient for mobile clouds with intermittent connection type
Client mobile agent	Usable even in an unstable network environment	Need to use network data for agent transmission
Server mobile agent	Possible to achieve fast processing speed	Limited to mobile agent services provided by the server

3. Mobile Resource Management without Cloud Server

In this section, Mobile Resource Management without Cloud Server (MRM) is introduced, which is a resource pool environment set up for mobile devices functioning only in the environment and with no cloud server or with poor connection in a mobile cloud computing structure and based on collaborative architecture. MRM can continuously use and manage mobile resources even in troubled situations, such as failure or disconnection of the cloud server. The mobile device that acts as a server in the mobile computing infrastructure plays the role of scheduler and controller for joint jobs. Figure 2a shows the basic conceptual structure of MRM, and Figure 2b shows three steps for MRM construction [21].

The MRM Configuration clusters connected mobile resources on a network, divides devices into client and master devices, and sets and activates master devices in line with their roles. The MRM Management continuously verifies the connections of mobile resources in the configured mobile cloud environment, and excludes disconnected resources or adds new resources. The MRM Service performs distributed processing for requested large-scale jobs and responds to problems in the client or master device. The MRM only uses the static and dynamic resources (MAC, IP, CPU, and memory) of the mobile devices. The static and dynamic metadata for mobile resources are outlined in Table 2.

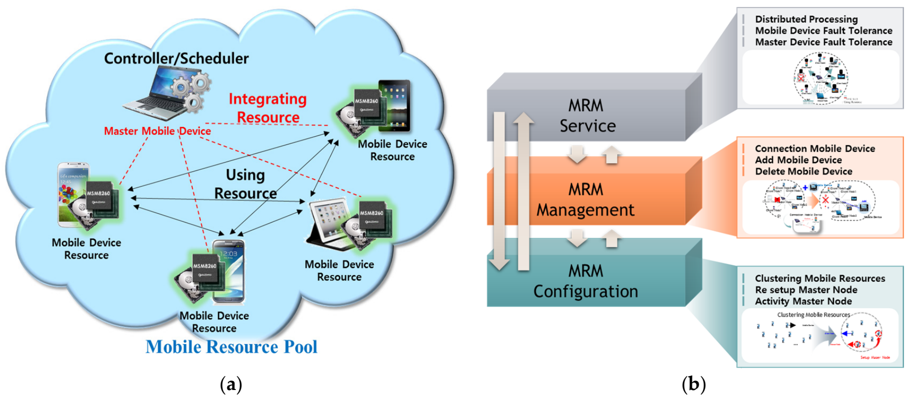


Figure 2. Basic architecture and three steps for mobile resource management construction. (**a**) Conception architecture of mobile resource management without cloud server (MRM). (**b**) Three steps for MRM construction.

Table 2. Mobile resource static and dynamic metadata.

Attributes	Static Resource Metadata	Dynamic Resource Metadata
MAC	MAC address of the corresponding client device	
IP	IP address of the corresponding client device	
CPU	Number of CPU cores of the corresponding client device	Current CPU share of the corresponding client device (%)
Memory	Memory size of the corresponding client device (MB)	Current memory share of the corresponding client device (%)

MRM uses the dynamic resource information for the mobile device at the time of distribution for dynamic job allocation. The processing steps are illustrated in Figure 3. As shown in Figure 3-①, a client device requests resources from a master device. As shown in Figure 3-②, the master device then creates job distribution metadata (JDM), which is a job distribution table based on the collected dynamic resource information. The configuration of JDM is outlined in Table 3.

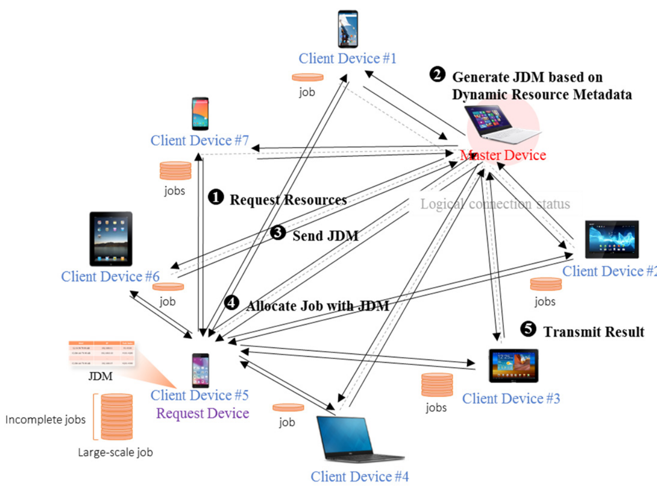


Figure 3. Load balancing with dynamic resource metadata for large-scale job. JDM: Job distribution metadata.

Table 3. Configuration of job distribution metadata.

Attributes	Description
MAC	MAC address of the client device to which a job has been assigned
IP	IP address of the client device to which a job has been assigned
Job Index	Index of the assigned job

Then, as shown in Figure 3-③, the JDM is sent to the device requesting the job, and, as shown in Figure 3-④, the job requesting device divides the job and sends a part of the job to each client device. Each client device processes the allocated job and returns the results to the job requesting device.

4. Adaptive Job Load Balancing Scheme

4.1. Execute Model

CPUs of the integrated mobile devices share CPU resources equally to perform multiple jobs. In this paper, the job execution time is calculated using the Equation (1).

$$Job_{Time} = \sum_{j=0}^{M}\left(\frac{\min\left(S_j \times (1 - U_j), \frac{\sum_{l=1}^{C}(S_j \times (1-U_j))}{C}\right)}{\sum_{i=0}^{Cd_{amount}} \min\left(S_j \times (1 - U_j), \frac{\sum_{l=1}^{C}(S_j \times (1-U_j))}{C}\right)} \times JobTotal\right) \tag{1}$$

The description of the symbols used in Equation (1) is provided in Table 4.

Table 4. Description of used symbols in Equation (1).

Symbols	Description
Job_{Time}	Completion time of large-scale job
M	Number of mobile devices in MRM
S_j	Static CPU capacity of mobile device j
U_j	CPU usage of mobile device j
C	30 s; Cycle of 30 s
$JobTotal$	Requested User Job

An adaptive large-scale job load balancing scheme is proposed for MRM functions considering the mobile characteristics where unexpected and sporadic personal user application jobs can be executed. The average CPU usage of mobile users is measured for a specific cycle of 30 s and is used as the average idle resource information for job distribution. The 30 s used as the measurement cycle for average CPU usage is the average usage time of smartphones per session of smartphone users, as identified in the research of Sally Andrews [22]. For information on the average idle resource, the AMRO scheme is used to improve large-scale job throughput by identifying the average resource used by users. The average idle resource information is defined as Average Idle Resource Information (AIRI), and the metadata is transmitted by adding the AIRI to the mobile resource information. For the master device, an Adaptive Resource Information Table (ARIT) is used in which the Dynamic Resource Information (DRI) and AIRI are collected, as opposed to the DRIT (Dynamic Resource Information Table) in which only DRI is collected. The equation for creating JDM using DRI and AIRI is expressed as follows:

$$DRI_i = S_j \times (1 - U_j) \tag{2}$$

$$AIRI_i = \frac{\sum_{l=1}^{C}(S_j \times (1 - U_j))}{C} \tag{3}$$

$$Job_i = \frac{\min_i(DRI_i, AIRI_i)}{\sum_{k=0}^{Cd_{amount}} \min_i(DRI_i, AIRI_i)} \times JobTotal \tag{4}$$

where Job_i denotes the number of jobs processed by the i^{th} mobile resource, Cd_{amount} denotes the total number of mobile resources configured, DRI_i denotes the dynamic idle resource information of the client device, $AIRI_i$ denotes the average idle resource information of the i^{th} client, $JobTotal$ denotes the total size of the large-scale job. This equation is used to select a value with fewer idle resources among the DRI and AIRI, and to divide the job accordingly.

When processing a large-scale job in a resource pool consisting of mobile resources only, the AMRO reallocates jobs according to the idle resource state of the device that is providing resources. If the processing of an offloaded job is delayed due to the execution of another application or for other reasons while a device is processing an allocated job, it is defined as a resource trouble. On the other hand, if an idle resource is generated because another application stops running while a device is running an allocated job, it is determined to be an idle resource. The equation for the criterion for determining an idle resource or a resource trouble is as follows:

$$ST_{second} = \frac{LargestJobDevice_{TotalJob}}{CurrentDevice_{Totaljob}} \times ProcessingTime_{FastJob} \tag{5}$$

where $LargestJobDevice_{TotalJob}$ denotes the total number of jobs of the mobile resources to which the largest number of jobs have been allocated, $CurrentDevice_{TotalJob}$ denotes the total number jobs that have been allocated to the current device, and $ProcessingTime_{FastJob}$ denotes the processing time of the job that has been processed fastest. A resource trouble occurs when the job processing speed of the current client device is slower than ST_{second}, and an idle resource occurs when it is faster than ST_{second}. To reduce unnecessary job reallocations due to the excessive determination of idle resources and resource troubles, a padding of $\pm 10\%$ is applied to the value obtained for ST_{second}.

4.2. Fault Tolerance Scheme

Figure 4 shows the adaptive processing when a resource trouble occurs in the mobile client while it is processing an allocated job. As shown in Figure 4-①, a resource trouble is detected when it is identified to the master device or when the master device receives real-time resource information. Figure 4-② shows the process of reallocating a job of a mobile client that has a resource trouble to an idle resource. A JDM of an incomplete job that has not been processed by the mobile client is created for a device that has an idle resource. As shown in Figure 4-③, the created JDM is sent to the job request device. The job request device, as shown in Figure 4-④, sends the incomplete job to a mobile client that has an idle resource. Figure 4-⑤ shows that the newly allocated incomplete jobs are processed by devices in reverse order. The device with a resource trouble processes jobs in the specified order and incomplete jobs in reverse order, thus processing the allocated jobs faster.

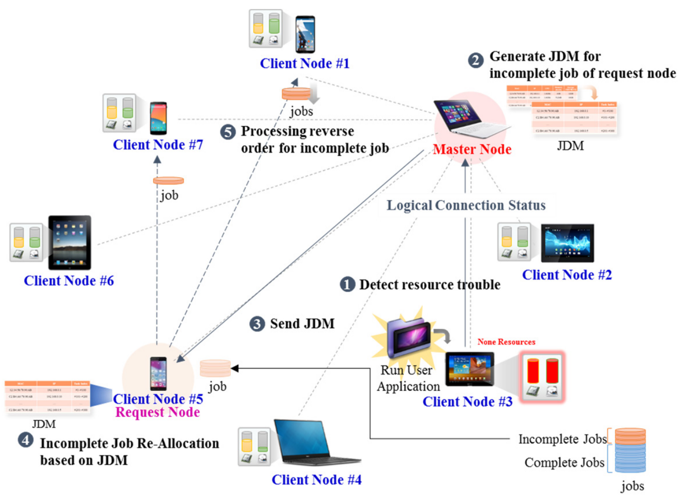


Figure 4. Fault tolerance scheme on adaptive mobile resource offloading (AMRO).

4.3. Idle Resource Scheme

Figure 5 shows the process when an idle resource occurs in the mobile client. In Figure 5-①, the master device detects an idle resource while collecting resource information from mobile clients. As shown in Figure 5-②, a JDM is created for an incomplete job based on the job processing status information that the master device receives. The JDM is sent to the job request device as shown in Figure 5-③, and the job request device sends the job to the mobile client in a resource idle state as shown in Figure 5-④. Figure 5-⑤ shows that the newly allocated incomplete jobs are processed in reverse order.

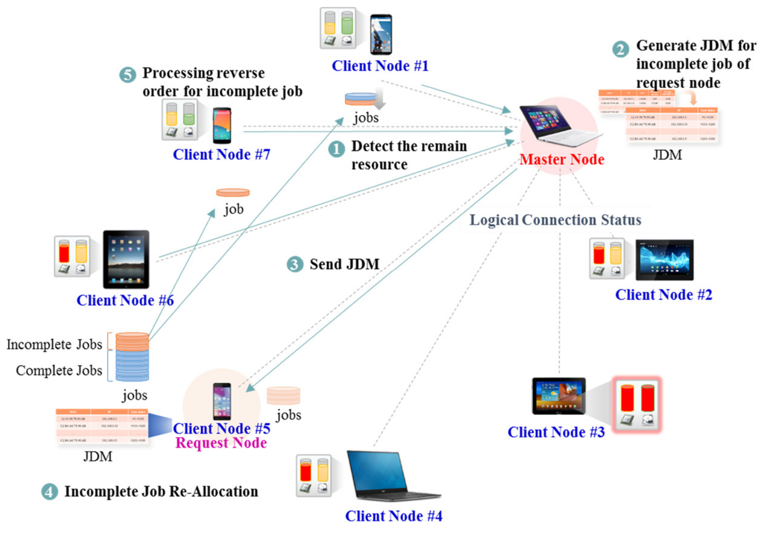


Figure 5. Idle resource scheme.

5. AMRM Design and Implementation

5.1. AMRM Design

In this section, the design of AMRM is described: this is a mobile cloud computing service with no server, which applies AMRO. When AMRO is applied to collaborative architecture using a high-performance cloud server, a visible comparison of performance may be difficult. Therefore, the AMRO was applied to the collaborative architecture with no server. Figure 6 shows the overall composition of the modules of AMRM.

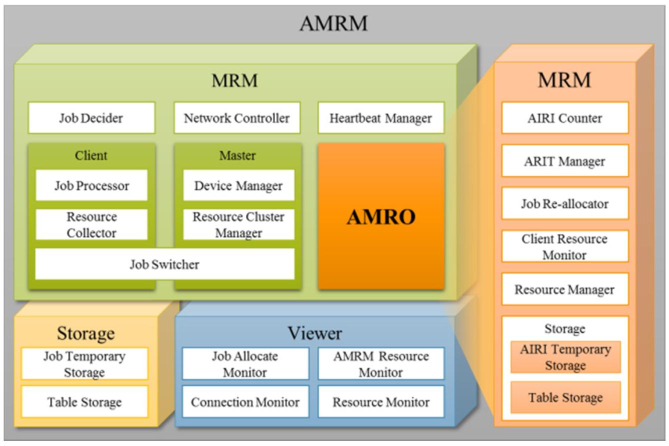


Figure 6. Module diagram of adaptive mobile resource management without cloud server (AMRM). MRM: Mobile resource management without cloud server.

Network Controller is responsible for the network connection of mobile devices and is used when mobile clients are connected to an initial master device. The Job Decider determines the roles of master and client according to their performance once the mobile cloud environment has been set up. After the devices share performance information, the Job Decider determines the master device which performs its allocated role. The master device and the mobile clients periodically communicate with each other through the Heartbeat Manager, which checks the occurrence of trouble on the basis of responses. The mobile client and the master device have client and master architectures, respectively, because they perform different jobs according to their specified roles.

Client is composed of a Job Processor and a Resource Collector. The Job Processor processes jobs sent from the resource request device. Every mobile client and master device shares the same Job Processor. In order to process different types of jobs, a new Job Processor must be distributed. The Resource Collector measures the idle resources of mobile devices and provides this information to the master device so that jobs can be allocated to idle resources.

Master is composed of the Device Manager and the Resource Cluster Manager. The Device Manager collects and manages information about the mobile clients in the cloud environment and copes with any problem that occurs among the mobile clients. The Resource Cluster Manager uses the mobile client information to determine the job allocation when it receives a job request from a mobile client.

Job Switcher is used when there is a problem in a mobile client or master device and their roles must be replaced. When a mobile device discovers a problem in the master device and determines that it has to play the role of master device based on the resource information, it finishes its client role and starts to play the master role through the Job Switcher.

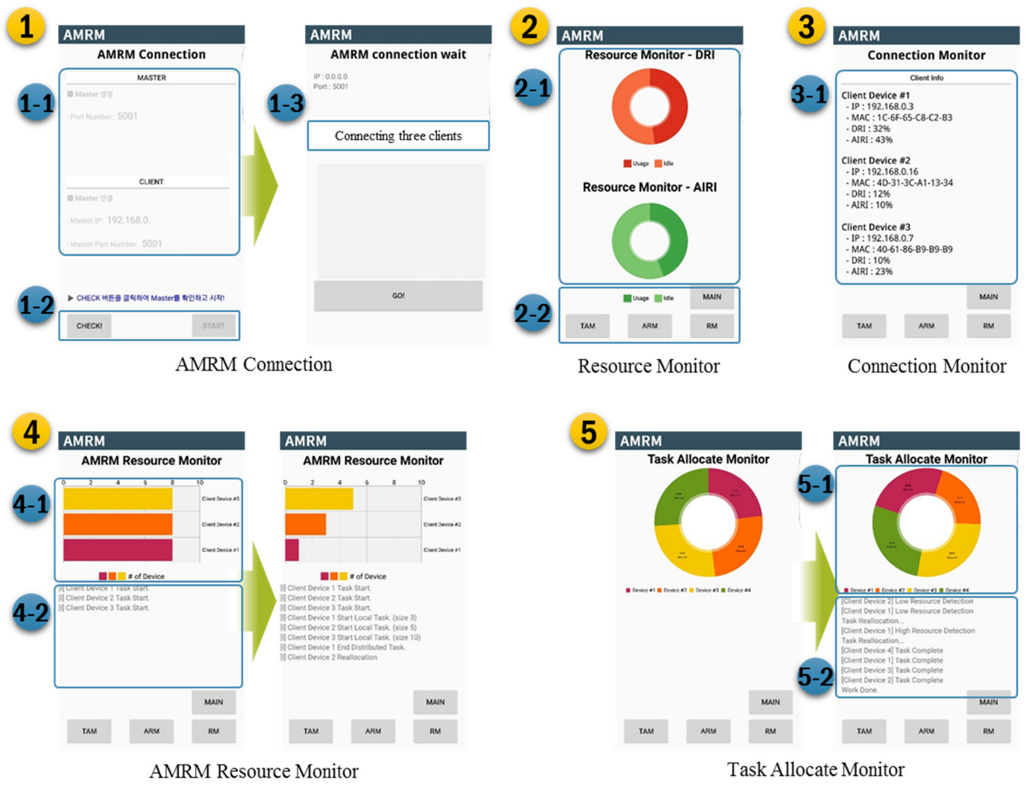
Storage is composed of Job Temporary Storage and Table Storage and is used to temporarily store jobs allocated to mobile clients or to store resource information. Job Temporary Storage receives and stores information about jobs to be processed by the mobile clients. The mobile client sequentially performs the jobs stored in Job Temporary Storage. Table Storage is used to temporarily store resource information and to table information about job distribution.

Viewer is composed of Job Allocate Monitor, AMRM Resource Monitor, Connection Monitor, and Resource Monitor. It is used to visually check the configuration, management, and services of the mobile cloud environment. Job Allocate Monitor is used to check the job allocation status through the AMRO, which makes it possible to check the adaptive offloading process when a problem occurs in a mobile client. The AMRM Resource Monitor shows the resources of the mobile clients as collected by the master device. It can see the status of jobs that are being processed by and are allocated to the mobile clients. In Connection Monitor, a list of the connection states of all mobile resources comprising the mobile cloud computing environment can be seen. The Resource Monitor shows all resources in the current mobile cloud, as well as the total resources that each mobile client can provide and the resources currently being used.

AMRO provides an adaptive mobile resource offloading scheme. The AIRI Counter calculates AIRI which is the average idle resource of the mobile device over 30 s. The ARIT Manager creates and manages the table ARIT, which integrates the AIRI received from the mobile clients with DRI. The Client Resource Monitor receives the AIRI of the mobile clients and determines the job processing speed. If the job processing speed is determined as fast or slow, jobs are reallocated through the Job Re-allocator. The Resource Manager configures the currently available resources considering the DRI and AIRI. The storage of AMRO consists of AIRI Temporary Storage and Table Storage. The AIRI Temporary Storage temporarily stores the AIRI received from each mobile client. The Table Storage temporarily stores the ARIT and JDM.

5.2. AMRM Implementation

Figure 7 shows the interfaces of AMRM as implemented on the Android platform, Java, and SDK Version 23.



```
Client Device #1
 - IP : 192.168.0.3
 - MAC : 1C-6F-65-C8-C2-B3
 - DRI : 32%
 - AIRI : 43%

Client Device #2
 - IP : 192.168.0.16
 - MAC : 4D-31-3C-A1-13-34
 - DRI : 12%
 - AIRI : 10%

Client Device #3
 - IP : 192.168.0.7
 - MAC : 40-61-86-B9-B9-B9
 - DRI : 10%
 - AIRI : 23%
```

```
[I] Client Device 1 Task Start.
[I] Client Device 2 Task Start.
[I] Client Device 3 Task Start.
[I] Client Device 1 Start Local Task. (size 3)
[I] Client Device 2 Start Local Task. (size 5)
[I] Client Device 3 Start Local Task. (size 10)
[I] Client Device 1 End Distributed Task.
[I] Client Device 2 Reallocation
```

```
[Client Device 2] Low Resource Detection
[Client Device 1] Low Resource Detection
Task Reallocation...
[Client Device 1] High Resource Detection
Task Reallocation...
[Client Device 4] Task Complete
[Client Device 1] Task Complete
[Client Device 3] Task Complete
[Client Device 2] Task Complete
Work Done.
```

Figure 7. Interfaces of AMRM.

Figure 7-① shows an interface for devices trying to access in order to configure the mobile cloud environment of AMRM. The steps are described in detail in the following:

Step 1-1 The devices start to access as master or client. The device that selected the master role waits for the access of mobile clients. The devices that selected the mobile client role enter their IP and the port of the master device and attempt to access the master device through the network.

Step 1-2 This interface consists of a Check button to check the mobile cloud configuration after selecting the master or client role and after entering the required information and clicking the Start button for connection.

Step 1-3 The master device checks the number of currently connected mobile clients. When the connection of mobile clients is completed, the mobile client configuration is finished.

Figure 7-② Here the Resource Monitor shows the available resources of the current mobile resource pool and the total resources after the mobile cloud environment of the AMRM has been set up.

Step 2-1 This interface shows a graph of DRI and AIRI for all mobile clients in the mobile cloud, as well as Usage, representing their current share and Idle, representing their idle resources.

Step 2-2 This interface shows menu buttons that can be used in the service environment of AMRM, which include Resource Monitor, Job Allocate Monitor, AMRM Resource Monitor, and Connection Monitor.

Figure 7-③ illustrates the Connection Monitor, which shows the real-time resource information of the mobile clients comprising the AMRM cloud environment.

Step 3-1 The information of connected mobile clients, which is updated according to the update cycle of resource information.

Figure 7-④ shows the state of job processing by each mobile client. The jobs are processed adaptively through the AMRO.

Step 4-1 This interface shows a graph of the number of jobs allocated to each mobile client. This graph decreases as the mobile client processes the jobs, and the jobs of devices that have a slow processing speed are reallocated to mobile clients that have already finished their jobs.

Step 4-2 This interface shows the results of the device state as determined by the AMRO and the reallocation results as logs.

Figure 7-⑤ provides progress information about job reallocation through the AMRO.

Step 5-1 This interface visually shows the total number of jobs allocated to each mobile client. The devices with faster processing speeds process more jobs through the AMRO.

Step 5-2 This interface shows the results of the device state as determined by the AMRO and the reallocation results as logs.

6. Performance Evaluation

To evaluate the performance of AMRO, the same jobs were performed using different offload methods in the MRM cloud environment and the time until job completion was measured. The relative performance of each offload method was evaluated based on the average of the measured times. The offload methods used in this performance evaluation were static resource information offload, dynamic resource information offload, and AMRO. The static resource information offload allocates jobs only according to the static information of the mobile devices. Static device information refers to SRI, that is, the CPU and RAM of the device. Performance changes by jobs are not considered. The dynamic resource information offload uses DRI at the time of the offload. Jobs are offloaded based on the current idle resources of the mobile devices. AMRO offloads jobs using both AIRI and DRI and adaptively reallocates jobs.

Figure 8 shows a comparison of cases where the mobile device does not run any other processes apart from providing resources to the mobile cloud computing. The times that elapsed during 100, 200, 500, and 1000 jobs in the same mobile cloud environment were measured. In experiments of 500-job and 1000-job processing, 100 and 200 jobs do not exhibit the result of static resource information in Figure 8. The measurements were divided into four units because job processing in a small unit cannot clearly portray the difference in time taken. The static resource information offload method for 500 and 1000 jobs was excluded from the verification because it took too much time compared with the other offload methods. The dynamic resource information offload and the AMRO showed similar processing times, but the static resource information offload was slower and had a less accurate performance measure as compared to the other two offload methods.

Figure 9 shows the processing time when the resource state changed due to the execution of a high-priority process in some mobile clients, which supplies resources three seconds after the job started. The executed process here creates a delay of three seconds regardless of the device performance. The times required for the processing of 100, 200, 500, and 1000 jobs were measured.

As shown Figure 9, the result of evaluation has the same experimental environments for the static resource information. Here AMRO showed a faster processing speed than the dynamic resource information offload through job reallocation, and the static resource information offload still showed the slowest processing speed.

In this research, AMRO evaluated the performance of random-allocation, static-allocation and dynamic-allocation according to increasing number of jobs as shown Figure 10 with 9 mobile devices on Android. Random-allocation decides the client node to allocate through random generation of a master node. As shown Figure 10, AMRO and dynamic-allocation exhibit the result of performance more than random/static-allocation. Additionally, AMRO is effective at about 59.32% for 100 jobs versus dynamic-allocation, about 33.66% for 200 jobs, about 32.78% for 500 jobs and about 7.58% for 1000 jobs, respectively.

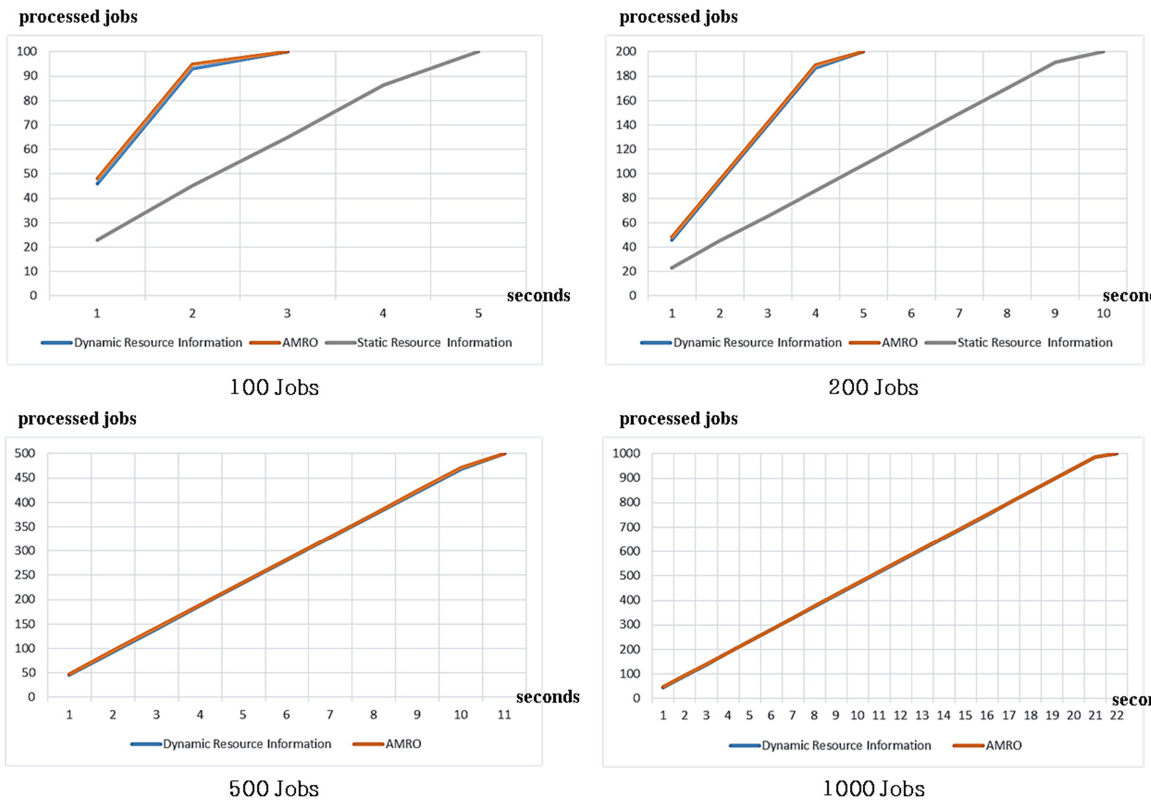


Figure 8. Job processing time with idle status resources.

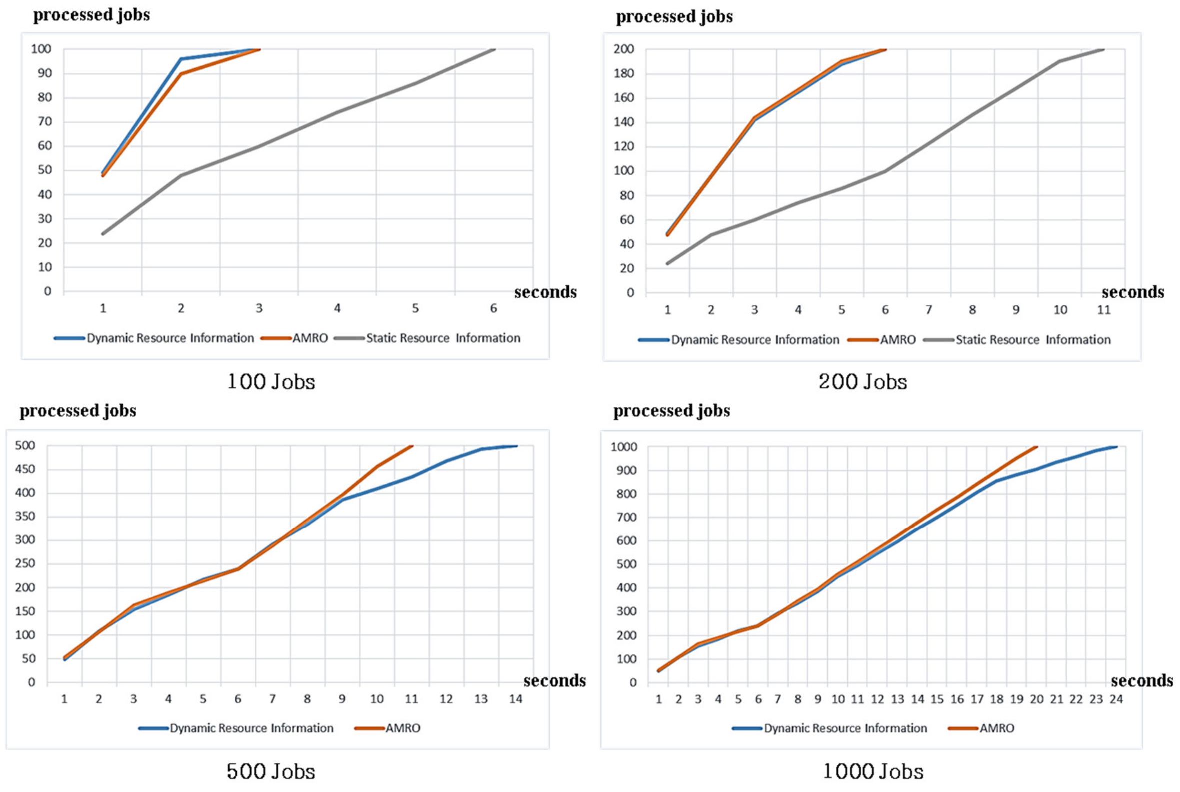


Figure 9. Job processing time with dynamic resource status.

For verification of our *execute model* of AMRO, 100 jobs was compared to the average value of simulated mobile devices in Figure 11. As shown Figure 11, AMRO is significantly faster than static-allocation and dynamic-allocation and the efficiency of actual execution environment is about 49.08% higher than simulation.

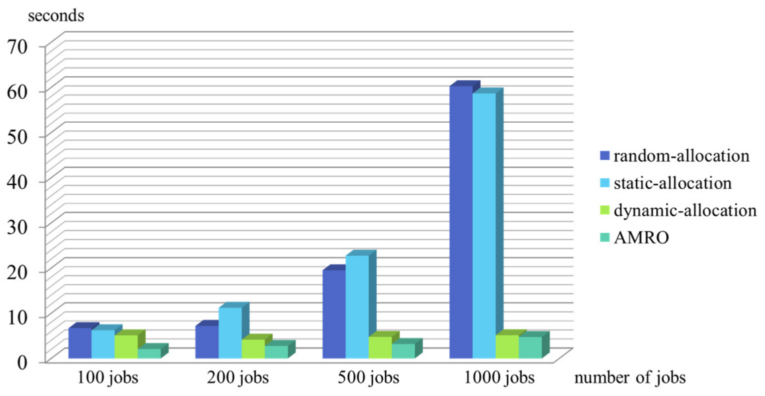


Figure 10. Comparisons according to increasing jobs of random-allocation, static-allocation, dynamic-allocation, and AMRO.

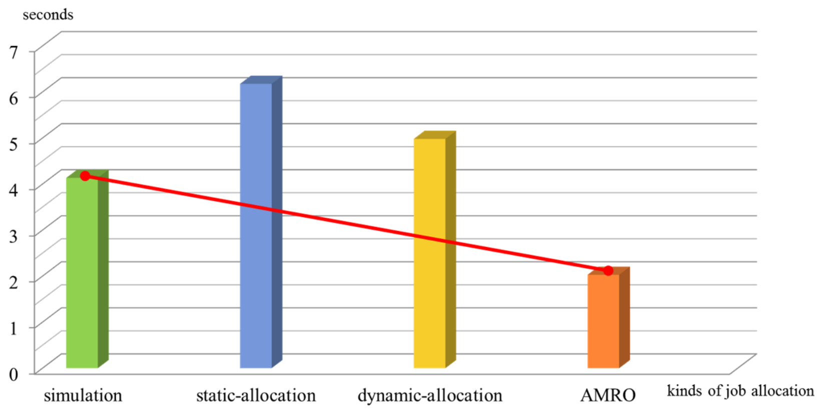


Figure 11. Comparison of random-allocation, static-allocation, dynamic-allocation, and AMRO.

7. Conclusions

This paper proposed an AMRO approach, which is an offload method that can respond to changes in the resource of mobile devices. For AMRO, an adaptive mobile resource offloading scheme was constructed to efficiently use the resources of mobile devices in the mobile cloud environment. The AMRM was designed and implemented to verify whether the proposed method had actually improved the processing speed compared to the existing offloading method. The AMRO increased the job processing speed in the mobile cloud environment through job allocation by considering the idle resources of mobile devices. Furthermore, the wastage of the idle resources of the mobile devices could be minimized by adaptive offloading through the identification of insufficient resources and idle resources. Thus, the AMRO offloading method was found to minimize the waste of resources in the mobile cloud environment when it consisted of mobile devices only with no cloud server. In the future, offloading methods for a mobile cloud environment with heterogeneous devices which includes mobile devices will be researched.

Acknowledgments: This work was supported by the Dongguk University Research Fund of 2016 (S-2016-G0001-00019) and was also supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2013-0-00684) supervised by the IITP (Institute for Information & communications Technology Promotion).

Author Contributions: All the authors contributed equally to this work. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sonkar, S.K.; Kharat, M.U. A survey on resource management in cloud computing environment. *Int. J. Adv. Trends Comput. Sci. Eng.* **2015**, *4*, 48–51.
2. Jennings, B.; Stadler, R. Resource management in clouds: survey and research challenges. *J. Netw. Syst. Manag.* **2015**, *23*, 567–619. [CrossRef]
3. Satyanarayanan, M. Fundamental challenges in mobile computing. In Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing (PODC '96), Philadelphia, PA, USA, 23–26 May 1996; pp. 1–7.
4. Mastelic, T.; Oleksiak, A.; Claussen, H.; Brandic, I.; Pierson, J.; Vasilakos, A.V. Cloud computing: Survey on energy efficiency. *ACM Comput. Surv.* **2015**, *47*, 1–36. [CrossRef]
5. Grozev, N.; Buyya, R. Inter-Cloud architectures and application brokering: Taxonomy and survey. *Softw. Pract. Exp.* **2014**, *44*, 369–390. [CrossRef]
6. Kashyap, D.; Viradiya, J. A survey of various load balancing algorithms in cloud computing. *Int. J. Sci. Technol. Res.* **2014**, *3*, 115–119.
7. Singh, A.; Kaur, I. A Survey on cloud computing and various scheduling algorithms. *Int. J. Adv. Res. Comput. Sci. Manag. Stud.* **2016**, *4*, 209–212.
8. Kulkarni, R.A.; Patil, S.H. A survey on improving performance of real time scheduling for cloud systems. *Int. J. Innov. Res. Sci. Technol.* **2015**, *1*, 171–173.
9. Fernando, N.; Loke, S.W.; Rahayu, W. Mobile cloud computing: A survey. *Future Gener. Comput. Syst.* **2013**, *29*, 84–106. [CrossRef]
10. Flinn, J.; Park, S.; Satyanarayanan, M. Balancing performance, energy, and quality in pervasive computing. In Proceedings of the 22nd International Conference on Distributed Computing Systems, Vienna, Austria, 2–5 July 2002; pp. 217–226.
11. Balan, R.; Satyanarayanan, M.; Park, S.; Okoshi, T. Tactics-based remote execution for mobile computing. In Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, San Francisco, CA, USA, 5–8 May 2003; pp. 273–286.
12. Kar, J.; Mishra, M.R. Mitigating threats and security metrics in cloud computing. *J. Inf. Process. Syst.* **2016**, *12*, 226–233.
13. Motavaselalhagh, F.; Esfahani, F.S.; Arabnia, H.R. Knowledge-based adaptable scheduler for SaaS providers in cloud computing. *Hum. Centric Comput. Inf. Sci.* **2015**, *5*, 1–19. [CrossRef]
14. Kim, S.; Lee, H.; Kwon, H.; Lee, S. Evaluation model of defense information systems use. *J. Converg.* **2015**, *6*, 18–26.
15. Park, J.H.; Kim, H.; Jeong, Y. Efficiency sustainability resource visual simulator for clustered desktop virtualization based on cloud infrastructure. *Sustainability* **2014**, *6*, 8079–8091. [CrossRef]
16. Kim, H.-W.; Park, J.H.; Jeong, Y.-S. Human-centric storage resource mechanism for big data on cloud service architecture. *J. Supercomput.* **2015**, *72*, 2437–2452. [CrossRef]
17. Kemp, R.; Palmer, N.; Kielmann, T.; Bal, H. Cuckoo: A computation offloading framework for smartphones. In Proceedings of the 4th International conference on Mobile Computing, Applications, and Services, Seattle, WA, USA, 11–12 October 2012; Volume 76, pp. 59–79.
18. Chun, B.-G.; Ihm, S.; Maniatis, P.; Naik, M.; Patti, A. Clonecloud: Elastic execution between mobile device and cloud. In Proceedings of the EuroSys 2011 Conference, Salzburg, Austria, 10–13 April 2011; pp. 301–314.
19. Clark, C.; Fraser, K.; Hand, S.; Hansen, J.; Jul, E.; Limpach, C.; Pratt, I.; Warfield, A. Live migration of virtual machines. In Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation (NSDI '05), Berkeley, CA, USA, 2–4 May 2005; pp. 273–286.

20. Kristensen, M. Scavenger: Transparent development of efficient cyber foraging applications. In Proceedings of the 2010 IEEE International Conference on Pervasive Computing and Communications, Mannheim, Germany, 29 March–2 April 2010; pp. 217–226.
21. Heo, Y.-A. A Study on Mobile Resource Management Scheme Based on Collaborative Architecture. Master's Thesis, Dongguk University, Seoul, Korea, January 2017.
22. Andrews, S.; Ellis, D.A.; Shaw, H.; Piwek, L. Beyond self-report: Tools to compare estimated and real-world smartphone use. *PLoS ONE* **2015**, *10*. [CrossRef] [PubMed]