Ali Al-ameri and Ihsan Alshahib Lami, "SCCOF: Smart Cooperative Computation Offloading Framework for Mobile Cloud Computing Services", ICT - BDCS 2017 (August 21 - 22, 2017). The published version is available under DOI: 10.5176/2251-2136_ICT-BDCS17.04.

SCCOF: Smart Cooperative Computation Offloading Framework for Mobile Cloud Computing Services

Ali Al-ameri and Ihsan Alshahib Lami
Applied Computing dept.
University of Buckingham
Buckingham, UK
e-mail: first.last@buckingham.ac.uk

***Abstract*— Virtual reality games and image processing Apps are examples of mobile cloud computing services (MCCS) common on Smartphones (SPs) nowadays, requiring intensive processing and/or wireless networking. The consequences are slow execution and huge battery consumption. Offloading the intensive computations of such Apps to a cloud based server can overcome such consequences. However, such offloading will introduce time delay and communication overheads. This paper proposes to do the offloading to nearby computing resources in a cooperative computation sharing network via short-range wireless connectivity. The proposed SCCOF reduces offloading response time and energy consumption overheads. SCCOF is supported by an intelligent cloud located controller that will form the cooperative resource sharing network on the go when needed, based on available devices in the vicinity, and will use the cloud if necessary. Upon the initiation of the MCCS service via the App, our controller will devise the offloaded VMs as well as the offloading network. A study test scenario was performed to evaluate the performance of SCCOF, resulting in saving of up to 16.2x in execution time and 57.25% energy.**

Keywords- mobile cloud computing services; smart cooperative computation offloading framework; virtual machines;

I. INTRODUCTION

As well as performing the necessary processing and communication for Apps such as social media, online gaming, banking, navigation and so on, Smartphones (SPs) can also be used for computing intensive mobile cloud computing services (MCCS) that require huge processing and wireless connectivity to cloud based servers. However, running these MCCS would result in draining the battery energy on the SP with rapid degradation of performance and eventually in a dead phone.

In recent years, there has been a rapid technology improvements of resources onboard SP/tablets/wearable devices/gadgets, including connectivity, memory, multi-core processing, display and battery capacity, and sensors with a plethora of wireless technologies. However, the results of a survey carried out in fifteen countries concluded that a longer battery lifetime is the main desired feature for SP subscribers to capitalize on using these resources to the full [1].

MCCS with offloading capability have been introduced to overcome SP limitations. These services merge the strength of cloud computing resources in terms of elasticity/flexibility and the convenience of SPs. Offloading refers to the concept of migrating the intensive computational tasks of such MCCS, bundled as a VM(s), from the SP to a server in the cloud to be processed before sending the results back to the SP in real time, as illustrated in Figure 1. A successful offloading would have to enhance the execution time and improve the battery consumption of SPs with Considerations of the amount of computation that needs to be offloaded, the cost of communication between the SP and the helping cloud servers, and the total delay to receive the results back, otherwise offloading is not beneficial. Ideally, offloading is most useful for tasks requiring large/complex computation that require few transfers between the SP and server done over a high bandwidth link such Wi-Fi/LTE.

Obviously the more offloading techniques improve, it will encourage more and more MCCS uptake by SP users, which will further encourage researchers to improve the offloading techniques and so on.

The main contributions of this paper are:

- Propose to do the offloading to nearby computing resources in a cooperative computation sharing network via short-range wireless connectivity.
- Introduce an on the go partitioning algorithm that can automatically distribute the App execution between the cooperative SPs.

The rest of this paper includes: Section II that summarizes recent offloading frameworks, while Section III presents the development of our proposed framework "SCCOF" and the steps of the on-the-go partitioning algorithm. Section IV presents the evaluation and analysis including offloading execution time and energy costs, a study test scenario to evaluate the performance of SCCOF. Conclusions and further work are in section V.

Gratitude to the University of Basra, and MOHESR (Ministry OF Higher Education and Scientific Research) for sponsoring this work.

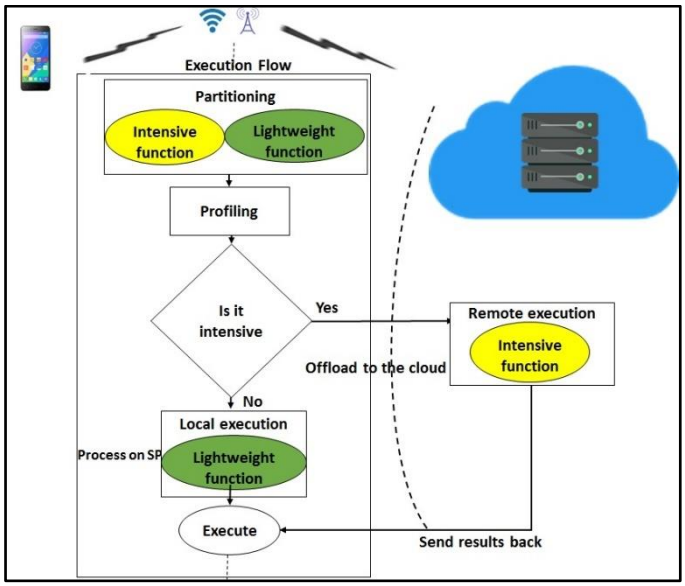

Figure 1. Classic Offloading

II. RECENT MOBILE-CLOUD OFFLOADING FRAMEWORKS

Elasticity and scalability of cloud computing resources with the help of offloading is promising to bridge the gap between limited resources on SPs and the growing demand of MCCS. One of the earliest frameworks that improved both SP execution time and energy consumption is Cuckoo [2]. It was developed to make the process of offloading easier for App developers by providing its functions in a dynamic runtime system. That is, at runtime, it divides an App into a local part that will run on the SP and a remote part to be offloaded to the cloud based on context information (availability of the resource). Two computationally intensive Apps, an object recognition, and a reality game, were used to test the framework. Although Cuckoo can enhance the execution time and reduce the energy consumption, but it based on a very simple context information algorithm, which means that it always offloads if the remote cloud server is available without consideration for communication and computation costs.

Another framework that aims to reduce the burden on App developers as well as to reduce SP energy consumption, but supports fine-grained code offloading is MAUI [3]. It achieves this by providing an integrated platform which runs partially between the SP and the server. MAUI consists of four components: (1) a profiler to gather information about the App, the SP platform, and the network; (2) a solver to get inputs from the profiler so to decide for each method the cost of offloading at runtime, including the amount of data to be transferred and the number of CPU cycles; (3) a proxy to handle the control and data transfer of the offloaded VMs; (4) a controller to determine the availability of the resources allocated. If the connection between the SP and the server is lost, then the proxy re-invokes the VM to run locally on the SP. Performing the processing computation of face detection by the cloud using a Wi-Fi connection have achieved up to 6.5x speed up and 89% energy saving compared to 4.75x and 76% when using 3G connection to offload. Although MAUI can reduce the burden on App developers and reduce energy consumption but the continuously running profilers that consume energy would increase the overhead by 5%.

In the other hand, CloneCloud framework focuses on selecting suitable VMs of the running App to offload from the host SP to their framework clone that operates in the cloud's server [4]. CloneCloud provides a partitioning mechanism which combines: (1) a static analyzer that identifies which part of the App to be offloaded based on a list of constraints, such methods which access SP features like GPS and Camera; (2) a dynamic profiler for the inputs of the SP and the clone; (3) both data collected from the analyzer and the profiler will then be sent to the solver that decides the portions to be offloaded. Performing the processing computation of image search App by the clone using Wi-Fi connection have achieved up to 20x speed up and 20% energy saving compared to 16x and 14% when using 3G connection for doing the offloading. Despite the fact that in case SP damage or lost, the clone can recover and backup data and Apps. However, the sync process between the SP and the clone in the cloud may increase the offloading overhead.

Note that, none of the above three frameworks provides a mechanism to support resource allocation "on demand". A framework (ThinkAir) that does consider the workload of the server in the cloud before offloading is developed to offer parallelism to multiple VMs while providing a dynamic on-demand resource allocation [5]. It supports dynamic adaptation based on the user requirement and on the workload, thus it can achieve more performance enhancement since parallelization can improve SP performance. It achieves this by providing: (1) a profiler to gather information of the App VMs that needs offloading, the network status, and the SP's platform capability; (2) a compiler to translates the annotated code and generates a remote VM; (3) a VM manager that can control the parallelization between VMs in the cloud and split tasks among VMs on demand. However, starting, resuming, and controlling VMs in the cloud can increase the latency by up to 32 seconds. This framework subsequently enhanced to improve the connectivity delay. That is, during the offloading process, if the network condition is poor due to (user movement or sudden drop in the network), no offloading is performed (delayed offloading) until the network status is improved for a predefined period of time [6]. Then the network condition is re-evaluated again to check if there is an improvement. If no improvement is applied to the network status, ThinkAir-enhanced resumes its offloading decision between the SP and the cloud. Their testing has shown that a considerable energy saving by up to 57% is achieved when performing the delayed offloading model.

Mobile-to-Mobile framework, However, shifts the offloading from the cloud to a nearby collaborative SPs in a wireless arrange which assumed to be in a home or a small office. Mobile-to-Mobile framework provides an auto-splitting algorithm that distributes the computation among offloader and offloadee SPs [7]. Their algorithm proposes that the moment the offloader SP initiates a task, the task will be classified and assigned a position in a queue waiting to be offloaded. Then based on a history database including previous execution time and energy, the decision will be made to execute on one of the Collaborative SPs. Performing the processing computation of

word count App by Mobile-to-Mobile framework have achieved up to 80% energy saving. Although their algorithm proves to distribute the computation between the collaborative SPs by up to 90% efficiency, but the algorithm is implemented on both offloader and offloadee SPs, and we believed this will also cost an extra energy overhead as well as delay in deciding when and where to offload.

The motivation for our framework is provide similar capability as Mobile-to-Mobile framework, but to enhance the delay of offloading and availability of resources from having the focus shifts from the cloud server into cooperative device in the vicinity of the SP. Furthermore, our proposed framework deploys a controller that performs the partitioning and profiling of VM(s).

The above reviews clearly show a story of continuous enhancement for the offloading technology. It starts with frameworks providing focus on developers to ease the offloading tasks. This then moved into enhancement made to ensure offloading done to an available server, followed by enhancement to assuring that connectivity is available and reliable. Our enhancement brings the server into locally available devices, thus reducing connectivity burden by default. We also shift the responsibility of partitioning to a cloud based controller, which also, as will be detailed later, acts as an intelligent divisor of local sharing network resources.

III. PROPOSED FRAMEORK ARCHITECHURE

SCCOF is unique in that it is an "on the go" framework that capitalizes on using available/cooperative local resources based on a cloud-based controller. As shown in Figure 2, SCCOF cloud-based controller logs/monitors all legacy task partitioning and offloading to any of the participated nearby SPs, so faster offloading execution is achieved. This controller also includes the partitioner function that will decide on the suitable VM(s) that will be executed on the SP and VM(s) that will be offloaded to neighboring devices based on many factors, including where the SP is and available nearby devices as well as issues to do with the App's intensive computation itself. This will enhance the SP available energy as well as eliminating delay in deciding when and where to offload, if done on the SP.

SCCOF's on-the-go algorithm will automatically splits the App execution, by also performing the partitioning/profiling based on the legacy data it has in its database collected from previous task offloading. It takes considerations of several layers of analysis and it depends on the circumstances where to offload and what is the overall achievement. The following steps explains the algorithm.

- The "Cooperative" offloading in SCCOF refers to a set of Devices (SP, tablet, PC, server) that are willing to participate in a cooperative environment by providing their local resources and share results with each other. Thus, each device has to register with the controller beforehand and has to exchange, via a secure key, a report including (location, memory level, battery level and processing capabilities), and whenever when asked by the controller.

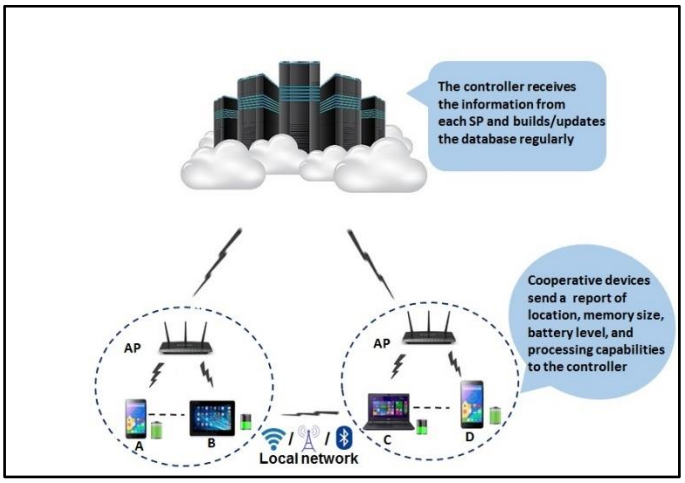

Figure 2. Proposed SCCOF architecture

- When the user has an App to execute, he/she interacts with SCCOF controller to help with the decision.
- App profiler to monitor the App and to generate a partitioning graph G (V, E) with vertex and edge, where V represents the task and E represents the relationship between task i to task j. It constructs a weighted consumption graph to find the computation and communication costs.
- SP profiler to profile SP local features such as battery level, CPU, and memory when running the App locally. There are two methods to profile SPs; either by using power monitor device attached to a SP or by using energy-wise software such as PowerTutor [8].
- Network profiler to gather information about the network type, connection status, and bandwidth when offloading.
- The partitioner receives the results from all the profilers (App, SP, and network), then it decides which part to run locally and the number and variety of VMs that can be offloaded.
- The scheduler assigns Id for each task and schedule tasks based on the availability of the participated devices and location.
- Then the controller takes the decision based on location, battery level, processing capabilities of neighboring cooperative SPs, cost of execution each specific VM and time, including communication costs of all offloadee SPs available to participate and especially if these devices are in the vicinity of the SP, where a fast/cheap communication link can be established (e.g. P2P Wi-Fi or BT). Once the controller performs all analysis/decisions, it will send the results to the SP. Furthermore, the controller builds/updates the database regularly for future executions.
- Then the user will communicate locally through (Wi-Fi, Cellular or Bluetooth) with the selected SPs, establish a session and request the results.

- Finally, the selected SPs will perform the requested computation and send the results back to the user.

IV. EVALUATION AND ANALYSIS

A. Offloading Execution Time and Energy Costs

A successful offloading would have to improve both SP execution time energy consumption when running processing intensive Apps. However, the offloading decision is critical and must be made based on the objectives; to improve SP execution time, save SP energy or both. Therefore, there need to be a careful consideration of many parameters such as the amount of computation that needs to be offloaded, the cost of communication between SP and the helping servers, and the total time to send the computation to the cloud, get it processed and receive the results back, otherwise offloading is not beneficial [9].

1. Improve execution time:

   The time needed to execute on the SP is:

$$Tp = \frac{w}{Sp} \quad (1)$$

   Where, Sp is the speed of SP (instructions per second), w is the amount of computation (number of instructions required by computation).

   So, the time needed to offload and execute on the server is:

$$Ts = \frac{D}{B} + \frac{w}{Ss} \quad (2)$$

   Where, Ss is the speed of the server (instructions per second), D is the amount of data (bytes), B is the network bandwidth (Kbps).

   Offloading improves the execution time only when the total time to execute on SP is greater than the total time required to execute on the server.

2. Save energy:

   The energy needed to perform the processing on SP is:

$$Ep = Pp \times \frac{w}{Sp} \quad (3)$$

   Where, Pp is the power consumption of SP (watts).

   So, the energy required to perform the computation on the server in the cloud and send the results back to SP is:

$$Es = Pc \times \frac{D}{B} + Pi \times \frac{w}{Ss} \quad (4)$$

   Where, Pc is the power consumption for sending and receiving data (watts), Pi is the idle power consumption (watts).

   Offloading reduces the total energy only when the energy spent by the SP is greater than the energy needed to send/receive and execute on the server in the cloud.

B. Study Test Scenario

For this study, the N-queens puzzle is chosen for being computationally requiring intensive processing. This puzzle is a strategy game for finding the suitable squares to place N queens on an N X N checkers board in squares.

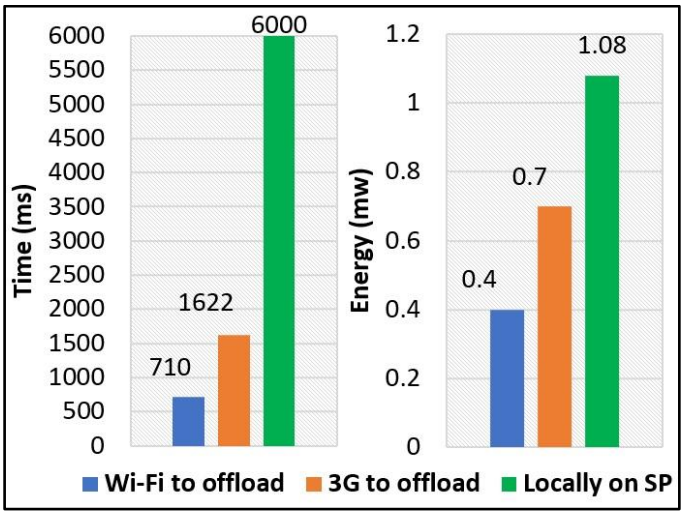

Figure 3. Execution time and energy consumption of N-queens puzzle

The basic rule of the N-queens is that queens will not threat each other vertically, horizontally, or diagonally. After analyzing N-queens, we found that It uses backtracking algorithm which is a high intensive search operation where it combines many intensive functions and loops such as (utility function to check if the queen can be placed on board, recursive function to solve row and column location and solver function to return the values). It tries to search for all possible solutions by going forward and backward until it reaches the correct solution. The solution can be formed in a binary matrix with a value of 1 if the queen is placed in a specific square and 0 if not. That is, running such puzzle on the SP would consume much-desired processing time and battery energy. When N increases, more iterations need to be found, for N=12, the algorithm will perform 14200 iterations to conclude the solution of placing the queens.

C. Results and Discussion

Figures 3, shows the average time and energy required to execute the N-queens puzzle locally on an SP, and in the cloud when connecting the SP to a server via Wi-Fi and via 3G. Using Wi-Fi connection to offload can execute in far less time compared to 3G connection and local execution, this is because 3G is slower than Wi-Fi and it has slower latency specially in congestion areas. Thus, this might be a bottleneck for offloading specially if Wi-Fi link between the SP and cloud is not available, and that is why SCCOF will also use Bluetooth to establish local connectivity with nearby SPs.

Furthermore, the energy required when using Wi-Fi and 3G connection to offload the computation to the cloud consume less energy compared to SP (locally), albeit, 3G results are still reasonable and promising to offload. The overall saving using Wi-Fi connection would achieve up to 16.27x speed up and 57.25% energy saving compared to 7.44x and 1.6% when using 3G connection to offload.

Since SCCOF can communicate with the cooperative offloadee SPs using Bluetooth, we did an experiment to calculate the transmission time of offloading a file size of 100 kilobytes.

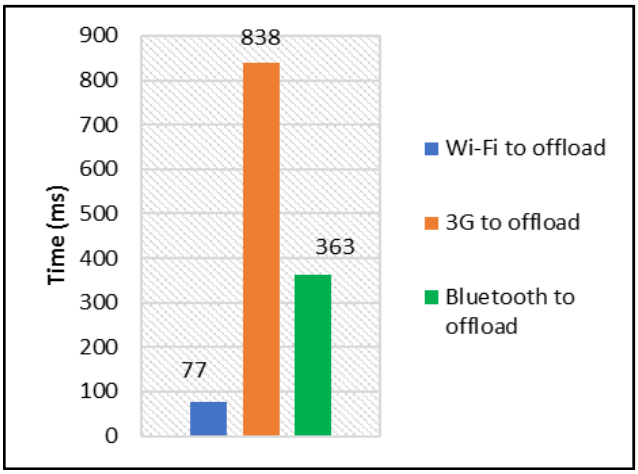

Figure 4. Data transmission time

2 iPhone 6 SPs are used in the experiment which act as offloader and offloadee SPs respectively, the SP is equipped with Dual-core 1.4 GHz Typhoon (ARM v8-based), 1 GB RAM DDR3 memory, battery capacity of Li-PO 1810 mAh (6.9wh) and it runs iOS 10.2. CalcTool, was used to calculate the transmission time a file size of 100 kilobytes form the offloader to the offloadee using Bluetooth with transfer rate of 2.1 MBPS. The results showed that it takes up to 363 milliseconds compared to 77 milliseconds for Wi-Fi and 838 milliseconds for 3G. Figure 4 shows the data transmission time between the offloader and offloadee device using of Wi-Fi, 3G and Bluetooth.

It is obvious that Bluetooth consumes far less time when offloading data from offloader to offloadee compared to 3G. Hence, we believe that SCCOF framework with the help of Bluetooth communication can reduce the time taken to transmit data to cloud/nearby SPs which would bring in advance more achievements to add to the offloading process.

V. CONCLUSION AND FURTHER WORK

SCCOF unique features to enhance published framework contributions have been discussed. However, it is worth reflecting here to point out issues we learnt thus far in this first-year work of a 3-year PhD programme.

SCCOF assumes, as fundamental to its performance, that there will be devices of various categories such as SPs, tablets, PC's on desktops and laptops available to participate in a resource-sharing network. We believe this is following on the trend to share anything to aid others, for gains that include (1) they will come to your help when you need to execute a computer intensive MCCs, and (2) a credit system can be devised between grouped-users that can be exchanged for monies or other sharing schemes.

Our proposed cloud-based controller has the potential to be very intelligent using deep learning engine and like. However, we believe that we will develop a simple machine learning controller and progress its development dependent on analysis of testing various scenarios and overhead costs for the offloading process.

Albeit, our study thus far is based on a simple scenario as proposed in IV, however we are preparing several offloading examples based on image processing/recognition in collaboration with colleagues in the department. Our study test scenario was performed to evaluate the performance of SCCOF, resulting in saving of up to 16.2x in execution time and 57.25% energy. We plan to extend our test scenario portfolio to include multiple offloading tasks at the same time onboard the same SP and using options of connectivity (Wi-Fi, Cellular/LTE and Bluetooth) in a changing dynamically environments (where connectivity is switched from one technology to another while offloading/execution). Note also that we will be using Bluetooth for short range connectivity which can be limiting, but Wi-Fi is also chosen at all times for p2p and WLAN connectivity.

REFERENCES

[1] S. M. Saad and S. C. Nandedkar, "Energy Efficient Mobile Cloud Computing," 2014.

[2] R. Kemp, N. Palmer, T. Kielmann and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in International Conference on Mobile Computing, Applications, and Services, 2010.

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra and P. Bahl, "MAUI: making smartphones last longer with code offload," in Proceedings of the 8th international conference on Mobile systems, applications, and services, 2010.

[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in Proceedings of the sixth conference on Computer systems, 2011.

[5] S. Kosta, A. Aucinas, P. Hui, R. Mortier and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in INFOCOM, 2012 Proceedings IEEE, 2012.

[6] M. Akram and A. ElNahas, "Energy-Aware Offloading Technique for Mobile Cloud Computing," in Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, 2015.

[7] G. Calice, A. Mtibaa, R. Beraldi and H. Alnuweiri, "Mobile-to-mobile opportunistic task splitting and offloading," in Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on, 2015.

[8] L. Z. a. B. T. Mark Gordon, "PowerTutor," University of Michigan, 17 November 2009.

[9] K. Kumar, J. Liu, Y.-H. Lu and B. Bhargava, "A survey of computation offloading for mobile systems," Mobile Networks and Applications, vol. 18, pp. 129-140, 2013.

ICT-BDCS Conference Secretariat [secretariat@bigdataclouds.org]

Actions
To:
ALI AL-AMERI; Ihsan Lami
Cc:
acceptance@globalstf.org
Attachments:
(2)Download all attachments
Copyright form.doc (48 KB)[Open as Web Page]; paper_ format.docx (31 KB)[Open as Web Page]

Thursday, April 20, 2017 8:36 AM

You replied on 4/21/2017 4:49 PM.

Dear Authors,

We are pleased to inform you that your full paper has been accepted for oral presentation ICT-BDCS 2017 in Singapore. Congratulations!

Please note that you are required to complete the registration where the instructions are available at conference website (visit http://bigdataclouds.org/Registration.html). Kindly note that the Early Registration Deadline is 22nd May 2017.

Please be informed that if the final version of your full paper exceeds the limit of 10 pages, charges will be incurred at SGD$50 per page. Hence, it is recommended that your paper is kept brief and succinct as possible.

Should you require any clarifications or assistance, please do not hesitate to contact us at secretariat@bigdataclouds.org

Briefly about the ICT-BDCS 2017 Conference:

- Keynote Addresses:
  - Prof. Yiu-ming Cheung, Department of Computer Science, Hong Kong Baptist University - *"How to Cluster Different Types of Data Towards Big Data Analytics?"*
  - Prof. Benjamin W. Wah, Professor of Computer Science and Engineering, Chinese University of Hong Kong - *"Using Kernels to Harness the Complexity of Big Data"*
  - Prof. Lynn Margaret Batten, Deakin Research Chair in Mathematics, School of Information Technology, Deakin University, Australia - *"Robust Trust Assessment Model for the Selection of Trustworthy Cloud Services"*
- ICT-BDCS 2017 Conference Proceedings All accepted and presented papers at the conference (LRPP 2017) will be published in the conference proceedings Print ISSN: 2251-3043, E-periodical: 2010-2283
- Journal: All authors who present their papers at the conference will be invited to submit an extended version of their research paper for the GSTF Journal on Computing (JoC) (Print ISSN: 2251-3043, E-periodical: 2010-2283) All submitted papers will go through blind review process for acceptance.
- Best Paper Awards and Best Student Paper Awards will be conferred at the conference (in order to qualify for the award, the paper must be presented at the conference).
- ICT-BDCS 2017 will also constitute a Special Panel Session.
- Panel Proposals are invited for submission. A minimum of three papers centering on a specific topic will be accepted for submission under Panel Category.

Should you have other questions, please feel free to contact us.

Warm Regards,

Ana Martina Tubilleja
Programme Manager
Global Science and Technology Forum
10 Anson Road, International Plaza,
#13-12, Singapore- 079903
Phone: +65 6327 0166| Fax: +65 6327 0162 | www.bigdataclouds.org
Co. Reg. No. 200923362W
www.globalstf.org