

Implementation of Crowd Sourced Work Stealing Mechanism

¹Prof. Sharmila Rathod, ²Shabnam Noorani

¹Professor, ²PG Student,

^{1,2}Dept. of Computer Engineering, Ragiv Gandhi College of Engineering,
Andheri, Maharashtra, India.

hbhor@live.com

Abstract - By pooling together the processing power of mobile devices within a crowd to form a 'mobile cloud', these devices be efficiently utilized to help realize the full potential of mobile computing. However, the dynamic nature of mobile computing makes sharing and coordinating work non-trivial. Although never been used before in the mobile computing domain, the concept of work stealing possesses useful traits such as self-adaptive ness, and decentralized nature that can help with these issues. Here we explore this concept of 'work stealing' for crowd computing on an opportunistic network of mobile devices, for both machine and human computation.

Keywords: *Mobile crowd computing, mobile cloud computing, work stealing.*

I. INTRODUCTION

Mobile computing can provide a computing tool when and where it is needed irrespective of user movement, thereby supporting location independence. However, the inherent problems of mobile computing such as resource scarcity, finite energy and low connectivity pose problems for most applications. These problems can be addressed by 'sharing' resource intensive work with a resource rich server. However in situations concerning mobile devices, connecting to a remote resource cloud via WiFi or 3G is not feasible because of bandwidth issues, data access fees, and the battery drain. Increasing usage and capabilities of smartphones, combined with the potential of crowd computing can provide a collaborative opportunistic resource pool to solve these problems. We define 'mobile crowd computing' as a local 'mobile resource cloud' comprising of a collection of local nearby mobile devices, utilized to achieve a common goal in a distributed manner.

In the Mobile Cloud computing environment, downloading speed will be affected due to the waiting for the request to be processed in the queue by the server. Without doing efficient assignment of the request to the servers, in some specific time, some servers will be in the overloaded status, some will be in the idle state. This causes the unnecessary delay in the downloading process. This issue is need to be solve by efficiently assigning the requests to the respective servers

based on the workload of the servers as well as work left to be done by the servers.

Crowd computing been conceptualized in various ways as being related to crowd sourcing, human computation, social computing, cloud computing and mobile computing[1][4]. A number of authors have put forward their own definitions to address a perceived lack of a common definition. Bessis et al. have noted that the emergence of differently labelled technologies with somewhat similar purposes can cause confusion, and in response have offered conceptions on how these technologies relate to each other. Schneider et al. have elaborated on earlier work and offer their own characterization of crowd computing systems, mapping out the application space that is encompassed by crowd computing. Some authors have referenced work on crowd computing by other researchers but it appears that the multiple streams of research and definitions[9] have evolved somewhat independently, perhaps due to the relative newness of this area of interest. By reviewing the extant literature on the subject of crowd computing, this paper aims to reconcile and integrate various descriptions that have been put forward in order to derive a definition of crowd computing. According to Pozzi, a definition serves to delimit an entity with respect to all others, and plays a central role in any inquiry[8]. A definition of crowd computing can be used to position the research already conducted on this subject to understand its relevance and coverage[5]. Therefore a clear definition is the starting point for further research, to

articulate a meaningful research problem to address gaps in prior research[3].

II. EXISTING SYSTEM

The existing work on victim selection in non-mobile computing domains such as suggest randomized and round robin selection methods and in a 'Pick-The-Richest' policy has been suggested. However, considering the need to conserve energy with the least amount of communication, we have decided against the 'Pick-The-Richest' policy. The stealing mechanism is initiated by a device (thief) sending a steal request to a potential victim. When a device receives such a transmission, depending on its available job queue, it can decide to become a victim[6][7]. The victim then removes a certain number of jobs from its own job list, and transmits them back to the thief. A device can be both a victim and a thief[1][4].

Disadvantages of the Existing System

- More processing power on a mobile device than on a node in distributed processing system.
- In a mobile cloud, the devices will be known to each other a priori, unlike in a grid environment.
- Heavy traffic when communicating the server.

III. PROPOSED SYSTEM

Work Stealing on multi processors. Each process maintains a double ended queue containing the jobs. Each process executes jobs from the head of the queue, and when the queue is empty, attempts to steal jobs from the tail of a queue that belongs to another process[3]. The concept of work stealing in the context of mobile cloud can be explained.

1. Video requests are sent to the Network gateway server from the N number of clients.
2. Network gateway server forwards the request to the Delegator Server.
3. Delegator Server receives the requests and creates the work chunks in the queue format.
4. If both Worker Server1 and Worker Server2 are in idle state, first work chunk will be assign to the Worker Server1 and second work chunk will be assign to the Worker Server2.
5. Worker Server1 and Worker Server2 download the videos from the Video Data Server with respect to the works chunks assigned to both.
6. If either any of the Worker Server completes the assigned works chunk, next work chunk will be steal from the Work Queue which is in Delegator Server.

After successful completion of all the works chunks both Worker Servers will go to the idle state[11].

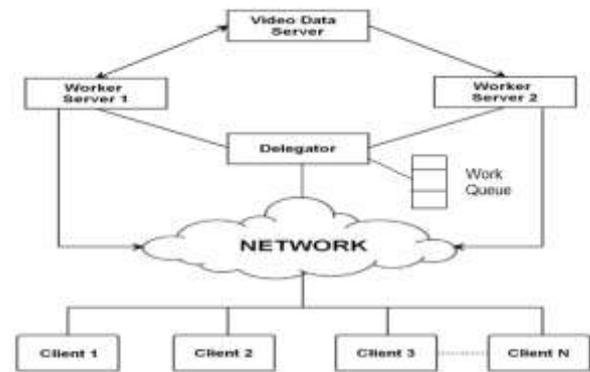


Fig. 1 System Architecture

Advantages of the Proposed System

- Less processing power on a mobile device than on a node in distributed processing system.
- A mobile node has on a finite energy source.
- A resource pool made up of mobile devices is highly volatile, and hence node availability is inconsistent.
- In a mobile cloud, the devices will be unknown to each other a priori, unlike in a grid environment.
- Where nodes are established and approved beforehand. Therefore a mobile cloud calls for a more opportunistic and ad hoc behavior.
- A mobile cloud is most likely to be heterogeneous.

Although 'work stealing' method has been employed for job scheduling with load balancing in distributed environments such as Cilk, and Parallel XML processing, it has not yet been used in mobile computing domain, as far as we know[2]. The need for dynamic load balancing was demonstrated in, where distributed Mandelbrot set generation was done over a set of mobile devices. Mandelbrot set generation is one of the two applications used here as well, but here, we use the work stealing mechanism to efficiently distribute tasks. Work stealing in a 'mobile cloud' would mean connecting opportunistically to unfamiliar devices, while considering the demands of connectivity on the limited battery as well. Therefore, our implementation employs an adjusted version of the traditional work stealing scheme to better suit mobile computing. We show that this mechanism will always give a speedup gain, provided the devices are in no great distance from each other[10].

IV. IMPLEMENTATION DETAILS

In this section we discuss results from applications employing work stealing on mobile cloud. The job distribution and coordination is done according to the work stealing mechanism. A worker device that can take video quickly, can travel to view the parade from different interest points, is able to finish his/her job list faster than others. Once a worker's job list is exhausted, he/she can send the video to the originating device, and has the option of

stealing video jobs from the delegator’s job list, and vice versa. Hence our approach of work stealing can be used to load balance jobs done via crowd-sourcing or human computation[12]. For example, a human who can take video faster would be on a higher skill level than one who is slower. The faster human can then, finish his/her job queue and steal more jobs from the delegator. In this scenario too, the system has no a priori knowledge of worker capabilities and work stealing method ensures the system adapts accordingly.

V. TESTING OF APPLICATION

A test case has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition. Test cases in this case is as shown in Table 1.

Table 1 Test Cases

Test Id	Test Input	Expected Result	Actual Result	Remarks
TES T_01	Enter valid name and password click on submit button in the admin login page	Admin should be navigated to homepage with the following options change password, audio category, video category, user details, server details	Admin is navigated to homepage with the following options change password, audio category, video category, user details, server details	Pass
TES T_02	Click on server details link in the menu bar	The server details that includes server no, server name, server ip, server port, total size, available	The server details that includes server no, server name, server ip, server port, total size,	Pass

		size must be shown	available size is shown	
TES T_03	Click on user details link in the menu bar	The user details that includes user id, user name, gender, address, city, email and phone no must be shown	The user details that includes user id, user name, gender ,address, city, email and phone no is shown	Pass
TES T_04	Click on video category link in the menu bar	The video category name and description must be shown	The video category name and description is shown	Pass
TES T_05	Click on change password link in the menu bar enter current and new password	A new password must be generated to the admin	A new password is generated to the admin	Pass
TES T_06	Click on upload audio in the menu bar in the user homepage choose the file and click on upload audio button	Confirmation message “audio uploaded successfully” must be shown	Confirmation message “audio uploaded successfully” is shown	Pass
TES T_07	Click on upload video in the menu bar in the user	Confirmation message “video uploaded successfully” must be	Confirmation message “video uploaded successfully” is shown	Pass

	homepage choose the file and click on upload video button	shown		
TES T_08	Click on audio file	The selected audio music must be played	The selected audio music is played	Pass
TES T_09	Click on my profile link in the menu bar	The user details that includes user id, contact no, home, address, city, email id must be shown	The user details that includes user id, contact no, home, address, city, email id is shown	Pass

Herewith we describes the data that was gathered from the numerous tests runs on the Mobile Crowd Computing with Work Stealing web application. This data is then analyzed, and some conclusions are drawn based upon the trends in the results. Each subsection displays graphs, which shows the performance of the existing and proposed system and is followed by a detailed analysis of each graph. Our results with work stealing on mobile devices show that it is a viable method for efficient work distribution in mobile cloud.

a) Performance of Work Allocation in Existing System

In the existing system, work will be assigned to the workers by the delegator in the alternative manner, by that workers will get the jobs before completion of the previously assigned jobs. This leads to the inefficient load balance to the worker, means worker may lack of work or it may overload at any instance of time. In the below graph, shows the jobs assigned to the worker1 and worker2 , worker1 load and the worker2 load difference is more because of alternatively assigning the jobs to the workers, this leads to the inefficient load balancing. Work allocation status of existing system of worker 1 and worker 2 shown in fig 2 given below.

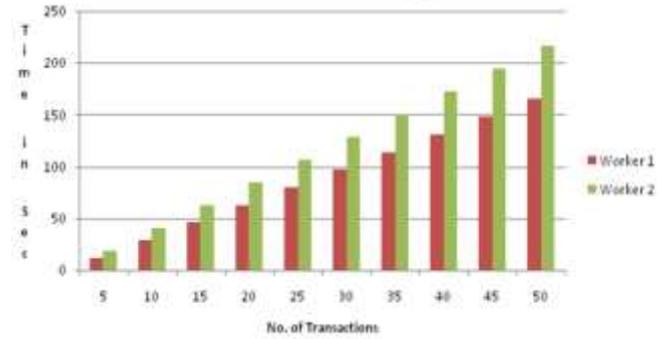


Figure 2 Work Allocations in Existing System

b. Performance of Crowd Computing with Work Stealing.

In the proposed system, jobs will be steal by the workers based on their work completion by those workers will get the jobs after completion of the previously assigned jobs. This leads to the efficient load balance to the worker, means worker will not lack of work or it will not overload at any instance of time. In the below graph, shows the jobs steal by the worker1 and worker2 , worker1 load and the worker2 load difference is less because of stealing the jobs by the workers after the completion of their previously assigned job, this leads to the efficient load balancing. Performance of Secure Crowd Computing with work Stealing of worker 1 and worker 2 shown in Fig. 3 given below.

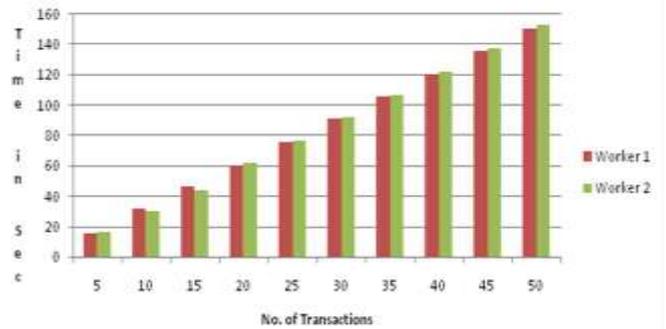


Figure 3 Crowd Computing with Work Stealing Performance

c) Result and Discussion

These results are valid for the ‘generative’ class of applications, where both the machine and human computation applications shown are ‘generative’ type, in that the job description is rather small, but the output results in a large amount of data that needs to be transmitted back. The goal for this project was to test the performance of a work Allocation in Existing System V/s Crowd Computing with Work Stealing on the Windows operating system. Performance analysis of the proposed system was performed by implementing the Crowd Computing work stealing as the web application. In our system we are using the two workers and one delegator. Based on the above mentioned resources, we tested and the result are presented in Result

Analysis section and we conclude that, The performance of the Crowd Computing with work stealing is more efficient and reliable than the existing system irrespective of number of workers.

VI. CONCLUSIONS

Our results with work stealing on mobile devices show that it is a viable method for efficient work distribution in a mobile cloud. We have demonstrated the possibility of a self-adaptive and decentralized mobile computation cloud that is able to obtain performance gains even without prior information about the participating devices. These results are valid for the 'generative' class of applications, where both the machine and human computation applications shown are 'generative' type, in that the job description is rather small, but the output results in a large amount of data that needs to be transmitted back.

VII. FUTURE SCOPE

Device participation is an important factor to the success of mobile crowd, and participation depends on the incentives. We hope to include incentive management in our framework in future work, where incentives could be in the form of social contract such as in a group of friends, common goals such as discussed in, or monetary as in the case of crowd sourcing done.

Although current experiments have involved only three devices this can further be scaled up to involve many more devices by implementing hierarchical stealing, where workers themselves become delegators. We aim to extend our implementation to use the Amazon cloud as well, since this would provide a comparison between offloading to local versus remote devices.

REFERENCES

- [1] Zeljko Vrba, Håvard Espeland, Pål Halvorsen, and Carsten Griwodz, "Limits of Work-Stealing Scheduling", E. Frachtenberg and U. Schwiegelshohn (Eds.): JSSPP 2009, LNCS 5798, pp. 280–299, 2009.
- [2] Wei Lu, Dennis Gannon, "Parallel XML Processing by Work Stealing", SOCP'07, June 26, 2007, Monterey, California, USA, ACM 978-1-59593-717-9/07/0006.
- [3] Niroshinie Fernando, Seng W. Loke and W. Rahayu, "Mobile Crowd Computing with Work Stealing", 15th International Conference on Network-Based Information Systems, 978-0-7695-4779-4/12, 2012 IEEE.
- [4] Prof. Sharmila Rathod, Shabnam Noorani, "Review of Mobile Crowd Computing", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 4, Issue 4, April 2016.
- [5] Derek G. Murray, Eiko Yoneki, Jon Crowcroft, Steven Hand, "The Case for Crowd Computing", MobiHeld 2010, August 30, 2010, New Delhi, India.
- [6] Paulo Mendes, Rute C Sofia, Jon Crowcroft, James Kempf, "User-Centric Networking", Dagstuhl Seminar 10372, September 12th-15th, 2010.
- [7] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu, Honeybee: A Programming Framework for Mobile Crowd Computing, MOBIQUITOUS 2013, LNCS 120, pp. 224–236, 2013.
- [8] Ana Luiza Dallora Moraes, Felipe Fonseca, Maria Gilda P. Esteves, Daniel Schneider, Jano M. de Souza, A Meta-Model for Crowdsourcing Platforms in Data Collection and Participatory Sensing, Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design.
- [9] Kalpana Parshotam, Crowd computing: A literature review and definition, Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, 2013.
- [10] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yuli Zhou. Cilk: an efficient multithreaded runtime system. *SIGPLAN Not.*, 30:207–216, August 1995.
- [11] Robert D. Blumofe and Charles E. Leiserson. Scheduling multithreaded computations by work stealing. *J. ACM*, 46(5):720–748, September 1999.
- [12] F. Warren Burton and M. Ronan Sleep. Executing functional programs on a virtual tree of processors. In *Proceedings of the 1981 conference on Functional programming languages and computer architecture*, FPCA '81, pages 187–194, New York,
- [13] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In Proceedings of the sixth conference on Computer systems, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.
- [14] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In Proceedings of the 8th international conference on Mobile systems, applications, and services, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.
- [15] James Dinan, D. Brian Larkins, P. Sadayappan, Sriram Krishnamoorthy, and Jarek Nieplocha. Scalable work stealing. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, pages 53:1–53:11, New York, NY, USA, 2009. ACM.
- [16] Daniel C Doolan, Sabin Tabirca, and Laurence T Yang. Mmpi a message passing interface for the mobile environment. In Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia, MoMM '08, pages 317–321, New York, NY, USA, 2008. ACM.
- [17] Gonzalo Huerta-Canepa and Dongman Lee. A virtual cloud computing provider for mobile devices. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS '10, pages 6:1–6:5, New York, NY, USA, 2010. ACM.
- [18] R Kemp, N Palmer, T Kielmann, and H Bal. Cuckoo: a computation offloading framework for smartphones. In Proceedings of The Second International Conference on Mobile Computing, Applications, and Services, MobiCASE '10, 2010.