International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2494-9150 Vol-02, Issue 07, Oct 2016

Crowd Sourced Work Stealing in Mobile Cloud Computing

[1]Prof. Sharmila Rathod, [2]Shabnam Noorani

[1]Professor, [2]PG Student, [1,2]Dept. of Computer Engineering, Ragiv Gandhi College of Engineering, Andheri, Maharashtra, India.

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. LITERATURE SURVEY

Niroshinie Fernando, Seng W. Loke and W. Rahayu proposed "Mobile Crowd Computing with Work Stealing". They explore concept of 'work stealing' for crowd computing on an opportunistic network of mobile devices, for both machine and human computation. They also present experimental data and discuss the findings. In paper they also explore results with work stealing on mobile devices show that it is a viable method for efficient work distribution in a mobile cloud. They 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[7].

Robert D. Blumofe Christopher F. Joerg Bradley C. Kuszmaul Charles E. Leiserson Keith H. Randall Yuli Zhou proposed concept of job scheduling with load balancing in distributed environments in paper "Cilk: An Efficient Multithreaded Runtime System". They discuss efficiency of the Cilk work-stealing scheduler, both empirically and analytically. They also show that on real and synthetic applications, the "work" and "critical path" of a Cilk computation can be used to accurately model performance[10].

Robert D. Blumofe proposed concept of scheduling Multithreaded Computation in paper "Scheduling Multithreaded computations by Work Stealing". They studied the problem of efficiently scheduling fully strict multithreaded computations on parallel computers. In this paper they discussed work staling scheduler for multithreaded computations with their dependencies[11].

Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, Ashwin Patti, discussed mobile cloud design & migration in paper "CloneCloud: Elastic Execution between Mobile Device and Cloud". This paper presents the design and implementation of CloneCloud, a system that automatically transforms mobile applications to benefit from the cloud. The system is a flexible application partitioner and execution runtime that enables unmodified mobile applications running in an application-level virtual machine to seamlessly off-load part of their execution from mobile devices onto device clones operating in a computational cloud. At runtime, the application partitioning is effected by migrating a thread from the mobile device at a chosen point to the clone in the cloud, executing there for the remainder of the partition, and re-integrating the migrated thread back to the mobile device[13].

Eduardo Cuervoy, Aruna Balasubramanianz, Dae-ki Cho, proposed VM migration in paper, "MAUI: Making Smartphones Last Longer with Code Offload". This paper presents MAUI, a system that enables fine-grained energy-aware offload of mobile code to the infrastructure. MAUI uses the benefits of a managed code environment to offer the best of both worlds: it supports fine-grained code offload to maximize energy savings with minimal burden on

the programmer. MAUI decides at runtime which methods should be remotely executed, driven by an optimization engine that achieves the best energy savings possible under the mobile device's current connectivity constrains[14].

James Dinan, D. Brian Larkins, P. Sadayappan, Sriram Krishnamoorthy, and Jarek Nieplocha proposed concept scalable work stealing in paper "Scalable work stealing". In this work author investigate the design and scalability of work stealing on modern distributed memory systems. We demonstrate high efficiency and low overhead when scaling to 8,192 processors for three benchmark codes: a producer-consumer benchmark, the unbalanced tree search benchmark, and a multiresolution analysis kernel[15].

Daniel C Doolan, Sabin Tabirca, and Laurence T Yang. put concept of message passing interface for the mobile environment. This paper looks at how all these devices can be used together of a collaborative fashion to solve parallel computing problems in a Java based environment. The means by which cross platform Bluetooth enabled applications can be developed and executed is examined, enabling mobile message passing to come into its own. The effectiveness of this parallel architecture is examined, with real world test results being presented to show that cross platform mobile parallel computing is more than a viable option[16].

N. Fernando, S.W. Loke, and W. Rahayu focuses concept of mobile cloud computing framework to use local resources in paper "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing". In this paper, Author explores the feasibility of a mobile cloud computing framework to use local resources to solve these problems. The framework aims to determine a priori the usefulness of sharing workload at runtime. The results of experiments conducted in Bluetooth transmission and an initial prototype are also presented. They also discuss a preliminary analytical model to determine whether or not a speedup will be possible in offloading[7].

Gonzalo Huerta-Canepa and Dongman Lee. discuss concept of A virtual cloud computing provider for mobile devices. In this paper, we present the motivation and preliminary design for a framework to create Ad Hoc cloud computing providers. This framework takes advantage of the pervasiveness of mobile devices, creating a cloud among the devices in the vicinity, allowing them to execute jobs between the devices[17].

R Kemp, N Palmer, T Kielmann, and H Bal. propose development of smartphone applications in paper "Cuckoo: a computation offloading framework for smartphones". In this paper authors present the first practical implementation of this idea for Android: the Cuckoo framework, which simplifies the development of smartphone applications that benefit from computation offloading and provides a dynamic runtime system, that can, at runtime, decide whether a part of an application will be executed locally or remotely[18].

Wei Lu and Dennis Gannon explores work on parallelism on the multicore machine the XML serialization in the paper "Parallel xml processing by work stealing". In this paper, Authors present a stealing-based parallel XML processing model. In this model the load balance among the threads is dynamically controlled by the stealing-based mechanism. And the stealing-based mechanism introduces only a little bit performance penalty, and this is mainly contributed by the stealing-from-bottom policy as well as the lock free ABP-deque. They show how the stealing-based mechanism , the stealing stub work and result gluing techniques be applied in the parallel XML serialization[2].

Derek G. Murray, Eiko Yoneki, Jon Crowcroft, and Steven Hand focuses the potential for crowd computing in the paper "The case for crowd computing". In this paper,

Author focuses the potential for crowd computing. Human interaction can be used to spread computation through an opportunistic network, and collect results. Furthermore, a simple task farming model can achieve reasonable performance in such a network, and dramatically better performance when community detection is used. A crowd computation should be energy-efficient, so the amount of wasted work must be small. Author also presented realistic model for crowd computing: static task farming[5].

III. 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 s 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.

IV. 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].

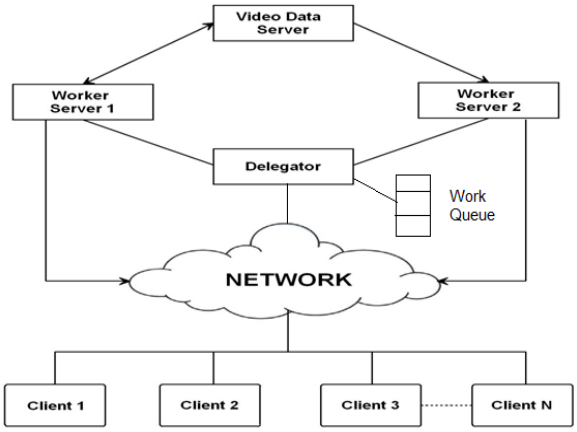

Fig. 1 System Architecture

Advantages of the Proposed System

- Less processing power on a mobile device than on s 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 before-hand. 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].

V. 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.

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, H˚avard Espeland, P˚al 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 HandThe, 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, LNICST 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.

