Efficient Resource Utilization Using Nearby Mobile Devices with Task Sharing Algorithm

[1]Deepak Oswal, [2]Prof.S.M.Sangve
Department of Computer Engineering, Zeal College of Engineering, Pune, India.

Abstract: *- In traditional web-based applications current technology does not facilitate exploiting this resource rich space of machine and human resources. As mobile devices evolve to be powerful and pervasive computing tools, their usage additionally continues to extend speedily. However, mobile device users oftentimes expertise issues once running intensive applications on the device itself, or offloading to remote clouds, attributable to resource shortage and property problems. Node heterogeneousness, unknown employee capability, and dynamism square measure identified as essential challenges to be self-addressed once programing work among near mobile devices we have a tendency to gift a work-sharing model, referred to as well-known work stealing methodology to load balance freelance jobs among heterogeneous mobile nodes, ready to accommodate nodes every which way effort and connection the system. The general strategy of this project is to specialize in short-term goals, taking advantage of opportunities as they arise, based on the ideas of proactive staff and timeserving delegator. We evaluate our model using a prototype framework built using Android and implement two applications.*

Keywords: - Wi-Fi, Hotspot, Job Scheduling, Load Balancing.

I. INTRODUCTION

Today's environments have become embedded with mobile devices with increased capabilities, equipped with numerous sensors, wireless connectivity also as restricted machine resources. However, on the far side some traditional web-based applications, current technology don't facilitate exploiting this resource wealthy house of machine and human resources. Collaboration among such sensible mobile devices will pave the Approach for larger computing opportunities, not simply by making crowd-sourced computing opportunities needing a person's component, however additionally by determination the resource
Limitation drawback inherent to mobile devices.

However such mobile crowds aren't meant to interchange the remote cloud computing model, however to enhance it as given below:

-As an alternate resource cloud in environments wherever connectivity to remote clouds is smallest.

-To decrease the strain on the network.

- To utilize machine resources of idle mobile devices [12].

This paper presents the Honeybee model that supports P2P work sharing among dynamic mobile nodes. As proof of concept we present the Honeybee API, a programming framework for developing mobile crowd computing applications. We build on previous work where we initially investigated static job farming among a heterogeneous group of mobile devices in [7], which was followed by a more self-adaptive approach in [6] using the 'work stealing' method and in [7] where three different mobile crowdsourcing applications were implemented and evaluated. The progress of our research on work sharing for mobile edge-clouds is illustrated in Table 1.

We present the honeybee model that supports P2P work sharing among dynamic mobile nodes. As proof of construct we have a tendency to gift the honeybee API, a programming framework for developing mobile crowd computing applications. we have a tendency to rest on previous work wherever we tend to at the start investigated static job farming among a heterogeneous cluster of mobile devices in, that was followed by an additional self-adaptive approach in using the 'work stealing' technique, and in wherever three completely different mobile crowdsourcing applications were enforced and evaluated. The progress of our analysis on work sharing for mobile edge-clouds is illustrated in Table 1.

Phase I	Phase II	Phase III
Simple work farming on Bluetooth	Work stealing on Bluetooth	Enhanced work stealing on Wi-Fi Direct: current paper
connect to workers via Bluetooth	connect to workers via Bluetooth	connect to workers via Wi-Fi Direct
distribute jobs equally	distribute jobs equally	work stealing commences without initial equal job distribution
No load-balancing	load-balancing via work stealing after initial job distribution	fault-tolerance and methods periodic resource discovery

TABLE 1: Evolution of the Honeybee model for computing with nearby mobile devices

We have improved the work stealing algorithmic rule of phase ii to deal with the bottlenecks within the transmission of enormous job information by optimizing the task distribution strategy and using Wi-Fi Direct. Phase III is additionally ready to handle random disconnections and opportunistic connections. We show wide amounts of performance gain and energy savings using our system. Though we tend to acknowledge that incentives, security and trust mechanisms are essential for a made mobile crowd, and honeybee is run on a secure atmosphere.

II. RELATED WORK

Offloading computation and storage from mobile devices to an external set of resources, has been explored in the literature [7]. With regards to the resource offloading, current research can be viewed from three main perspectives: offloading to a remote resource cloud [9], to a local cloudlet or local infrastructure [12] and to other mobile devices [7]. Each of the three methods have advantages depending mainly on the existence of high connectivity, additional infrastructure or node encounters respectively. In our work, we focus on the third method, ie., opportunistically sharing work with the surrounding mobile devices, owing to issues with the other two approaches in cases of low network availability and lack of established infrastructure. Furthermore, in Honeybee, we also recognize the potential of using mobile devices as agents of crowd sourcing, thereby exploiting the collective power of human expertise and machine resources.

In much research regarding mobile work sharing, the existence of a central server has been essential to either co-ordinate jobs among the mobile devices [10], or to offload the work on to [2], [3], [9]. However, our system follows a decentralized job sharing method, with the job scheduling depending entirely on the availability of the participating nodes. The concept of mobile devices forming resource clouds has been discussed by Miluzzo et al. in, which identifies key areas of 'MCloud Management' including periodic resource discovery, formation, fault tolerance, and handling mobility. In Honeybee, we also recognize the need to address the aforementioned areas, plus load balancing, and provide a complete implementation that supports them. An emulation testbed to evaluate the time and energy savings of offloading to a Mobile Device Cloud has been implemented in [6]. Such a testbed can be useful for mobile application development using an API such as Honeybee and some of the results reported from their testbed are comparable with our figures. However, our experimental data also suggest that there are additional factors that affect the overall performance such as accommodating random disconnections, unknown node capabilities, and unequal job distributions. Phoenix [11] proposes a distributed storage service using mobile devices in the vicinity, and shows the possibility to ensure data longevity despite autonomous node mobility. Honeybee, on the other hand, focuses on offering computation services rather than storage. In most mobile task sharing systems, Wi-Fi or 3G has been the most used communication protocols, except in the cases such as the MMPI framework [5], which is a mobile version of the standard MPI over Bluetooth, and uses Bluetooth exclusively for transmission, and Cuckoo [9], based on the Ibis communication middleware [13], to offload to a remote resource, and supports Bluetooth with Wi-Fi and cellular. Although Honeybee has used Bluetooth in previous versions, the current implementation uses Wi-Fi Direct due to better speeds and range. Femto Cloud [8] proposes an opportunistic mobile edge-cloud platform that offloads jobs to nearby mobiles, similarly to Honeybee. However, whereas Honeybee does not require prior information about the computational capabilities of the worker nodes to load balance the task, Femto Cloud's scheduling strategy depends on periodic capability estimations of each worker node.

III. MODEL AND ALGORITHMS

We define Mobile Crowd Computing as a bunch of dynamically connected mobile devices and their users using their combined machine and human intelligence to execute a task in a distributed manner. Such a mobile crowd is comprised of heterogeneous devices and will be unknown to every alternative a priori. Taking part mobile nodes could dynamically leave or be a part of the crowd while not prior notice, and therefore the should be accommodated by opportunistically seeking out new resources as they're encountered and having acceptable fault-tolerance mechanisms to support mobility.

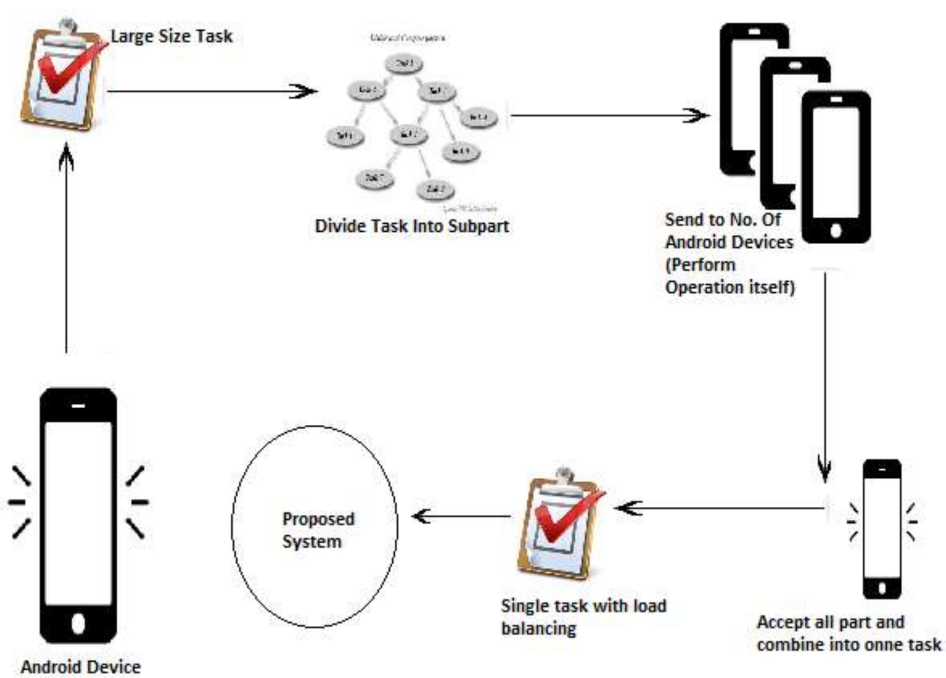

Figure 1: Architecture Diagram

Honeybee accommodates the higher than needs by being proactive and opportunistic, wherever jobs are 'taken' by nodes instead of 'given to' nodes, because the accessibility and resourcefulness of every node is unknown a priority, and subject to change any time.

3.1 Job Scheduling Method

The following characteristics of a mobile edge-cloud need to be considered when scheduling jobs among nodes:

1. Heterogeneity: since nodes could also be of heterogeneous capability and jobs could need varied amounts of resources, job allocation is non-trivial. Optimally stronger nodes should do additional work. An expiration mechanism is required so stronger nodes will steal terminated jobs taken by weaker nodes. Otherwise, if jobs were farmed equally, weak nodes could become bottlenecks.
2. Unknown capability: since the delegator is unaware of worker capability, it's impractical for the delegator to assign additional work to stronger nodes. Exchanging information isn't effective thanks to node dynamism, e.g., the node capabilities could modification randomly, thereby creating the knowledge derived from Meta data invalid.
3. Dynamism: as a result of mobility and factors like human intervention and low battery, nodes ar at risk of failure. Thus the likelihood of oftentimes disconnections and new nodes at random change of integrity need to be supported, and also the overall strategy must concentrate on short term goals and take advantage of opportunities as they arise.

Algorithm 1 : Job Scheduling Using Honey bee behavior inspired load balancing (HBB_LB) Algorithm.

Input : divided image chunks.
Output : Processed chunks
Step 1:- Get the available mobile resources from .ie, M1, M2… Mm

Step 2:- Submit the list of tasks T=T1, T2…Tn by the user.

Assign those task to available machines. When one of the machine get complete their job then fallow following procedure.

Step 3:- The scheduler finds the Expected computing capacity for tasks using (mbps) is million bits per second and (n) is the total number of tasks.

ECC=(mbps/n)

Step 4:- Compute the average computing capacity for each task using the equation,

ACC=(1/m)*ECC
m: Number of Ms
Step 5:- Find the load on a M

LM=(tasklength/servicelenth)

Step 6:- Compute the average system load

ASL=(1/M)*LM

Step 7:- The deviation of Load, DOL is found out as,

DOL=(ASL-LM)

Step 7.1 The probability value is checked for confinement within the range 0 to 1 as,

If (0< P(DOL)<1)

Underloaded_list[]= M

else

Overloaded_list[]= M

Step 8:- Select Underloaded Ms and compare its Average computing capacity with expected computing power of tasks.

Step 8.1 Check if (ACC< =ECC), then

Ms are marked as Fittest and tasks are allocated to it.

This is for all underloaded list of Ms First. When underloaded Ms list while(m!=0)the go for step9.

Step 9:- after task allocation to Ms, some Ms remains underutilized.

Check if (ACC>ECC), then

VMs are marked as weak and tasks are allocated to it.

Step 10:- If one of the M complete their job and return back to server then again go for step no-5 until your all task could not get complete.

IV. CONCLUSION

Firstly, work sharing among associate degree autonomous native mobile device crowd could be a viable technique to attain speedups and save energy. The addition of latest resources up to associate degree optimum quantity will yield inflated speedups and power savings. Secondly, generalized frameworks are often used for abstracting ways and facultative parameterization for various varieties of tasks product of freelance jobs. Thirdly, inherent challenges of mobile computing like random disconnections, having no previous info on taking part nodes, and frequent fluctuations in resource convenience are often with success accommodated via fault tolerance ways and work stealing mechanisms.

REFERENCES

[1]Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu Computing with Nearby Mobile Devices: a Work Sharing Algorithm for Mobile Edge-Clouds DOI 10.1109/TCC.2016.2560163, IEEE Transactions on Cloud Computing

[2] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. In Proc. of the 6th conference on Computer systems, EuroSys, pages 301–314, 2011.

[3] E. Cuervo, A. Balasubramanian, Dae-ki Cho, A.Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In Proc. of the 8th Intl conference on Mobile systems, applications, and services, MobiSys, pages 49–62, New York, USA, 2010. ACM.

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing, 2011.

[5] D. C. Doolan, S. Tabirca, and L. T. Yang. Mobile parallel computing. In Proc. of the 5th Int'l Symposium on Parallel and Distributed Computing, pages 161–167, 2006.

[6] A. Fahim, A. Mtibaa, and K. A. Harras. Making the case for computational offloading in mobile device clouds. In Proc. of the 19th Int'l Conference on Mobile Computing & Networking, pages 203–205, NY, USA, 2013.

[7] N. Fernando, S. W. Loke, and W. Rahayu. Honeybee: A programming framework for mobile crowd computing. In Mobile and Ubiquitous Systems: Computing, Networking, and Services, volume 120, pages 224–236. Springer Berlin Heidelberg, 2013.

[8] K. Habak, M. Ammar, K. Harras, and E. Zegura. Femtoclouds: Leveraging mobile devices to provide cloud service at the edge. In Proceedings of the 8th IEEE International Conference on CloudComputing, 2015.

[9] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Cuckoo: A computation offloading framework for smartphones. In Mobile Computing, Applications, and Services, volume 76, pages 59–79. Springer Berlin Heidelberg, 2012.

[10] E. E. Marinelli. Hyrax: Cloud Computing on Mobile Devices using MapReduce. Carnegie Mellon University, Masters thesis, 2009.

[11] R.K. Panta, R. Jana, F. Cheng, Y.R. Chen, and V.A. Vaishampayan. Phoenix: Storage using an autonomous mobile infrastructure. Parallel and Distributed Systems, IEEE Transactions on, 24(9):1863–1873, 2013.

[12] R. van Nieuwpoort, J. Maassen, G. Wrzesi ´ nska, R. F. H. Hofman, C. J. H. Jacobs, T. Kielmann, and H. E. Bal. Ibis: A flexible and efficient java-based grid programming environment: Research articles. Concurr. Comput. : Pract. Exper., 17(7-8):1079–1107, June 2005..