COMPUTING WITH NEARBY MOBILE DEVICES: A WORK SHARING ALGORITHM FOR MOBILE EDGE-CLOUDS

E.santhosh[1],V.Anjali[2]

[1]*Dept of Computer Science, School of Arts and Science, Vinayaka Mission's Research Foundation, Chennai, India.*
Email : santhoshvin123@gmail.com

[2]*Asst Professor, Dept of Computer Science, School of Arts and Science, Vinayaka Mission's Research Foundation, Chennai, India.*
Email: Anjali.mca@avit.ac.in

ABSTRACT

As mobile devices evolve to be powerful and pervasive computing tools, their usage also continues to increase rapidly. However, mobile device users frequently experience problems when running intensive applications on the device itself, or offloading to remote clouds, due to resource shortage and connectivity issues. Ironically, most users' environments are saturated with devices with significant computational resources. This paper argues that nearby mobile devices can efficiently be utilised as a crowd-powered resource cloud to complement the remote clouds. Node heterogeneity, unknown worker capability, and dynamism are identified as essential challenges to be addressed when scheduling work among nearby mobile devices. We present a work sharing model, called Honeybee, using an adaptation of the well-known work stealing method to load balance independent jobs among heterogeneous mobile nodes, able to accommodate nodes randomly leaving and joining the system. The overall strategy of Honeybee is to focus on short-term goals, taking advantage of opportunities as they arise, based on the concepts of proactive workers and opportunistic delegator. We evaluate our model using a prototype framework built using Android and implement two applications. We report speedups of up to 4 with seven devices and energy savings up to 71% with eight devices.

Keywords: *Mobile edge-clouds, crowdsourcing, mobile crowd computing, offloading*

I. INTRODUCTION

Today's environments are becoming embedded with mobile devices with augmented capabilities, equipped with various sensors, wireless connectivity as well as limited computational resources. Whether we are on the move, on a train, or at an airport, in a shopping centre or on a bus, a plethora of mobile devices surround us every day [47], thus creating a resource-saturated ecosystem of machine and human intel- ligence. However, beyond some traditional web-based ap- plications, current technology does not facilitate exploiting this resource rich space of machine and human resources. Collaboration among such smart mobile devices can pave the way for greater computing opportunities [54], not just by by creating crowd-sourced computing opportunities [29] needing a human element, but also by solving the resource limitation problem inherent to mobile devices. The focus of this paper is on mobile crowd (or edge-cloud). In our view, the human user of a mobile device is also a resource, which adds an element of crowd computing [48] to the mobile cloud as well. Therefore, we refer to this specialized mobile cloud as the Mobile Crowd. There are several unique features that differentiate mobile crowd environments from a typical grid/distributed computing cluster, such as less computation power and limited energy on nodes, node mobility resulting in frequent disconnections, and node heterogeneity [22]. Hence, solutions from grid/distributed computing cannot be used as they

are, and need to be adapted to suit the requirements of mobile crowd environments.

Related Works

Offloading computation and storage from mobile devices to an external set of resources, has been explored in the 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 [18], which is a mobile version of the standard MPI over Bluetooth, and uses Bluetooth exclusively for transmission, and Cuckoo[34], based on the Ibis communication middleware [62], to offload to a remote resource, and supports Bluetooth with Wi-Fi and cellular. Although Honeybee has used Bluetooth in previ- ous versions, the current implementation uses Wi-Fi Direct due to better speeds and range. FemtoCloud [28] 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, FemtoCloud's scheduling strategy depends on periodic capability estimations of each worker node. At the other end of the spectrum, crowd computing[48], [47], [52] has been shown to have the potential to use mobile devices in a social context to perform large scale distributed computations, via a static farming method. However, our results show that the work stealing method can provide better results. Social aware task farming has been proposed as an improvement on simple task farming, and social aware algorithms show better performance in their simulation based on real world human encounter traces [48]. In the future we hope to build on this result (social aware task sharing) as an incentive for participation. In [26], human ex- pertise is used to answer queries that prove too complicated for search engines and database systems, and in Crowd- Search [64], image search on mobile devices is performed with human validation via Amazon Mechanical Turk. A generic spatial crowdsourcing platform using smartphones is discussed in [11], where queries are based on location information. Mobile phones are used to collect sensor data on Medusa [53], according to sensing tasks specified by users. In Rankr [41], an online mobile service is used to ask users to rank ideas and photos. These are primarily con- cerned with the crowdsourcing aspect, using mobile devices as tools to access an online crowdsourcing service that is hosted on a remote server. In contrast, Honeybee defines the crowd as the surrounding mobile devices and their users, and focuses on sharing the tasks on a crowd of local mobile devices with performance gain and saving energy as the main goal. Indeed, results from the above research show us that user participation is at a considerable level, and using micro payments for such 'micro tasks' is viable.

II. EXISTING SYSTEM

Cloud computing is efficient and scalable but maintaining the stability of processing so many jobs in the cloud computing environment is a very complex problem with load balancing receiving much attention for researchers. Since the job arrival pattern is not predictable and the capacities of each node in the cloud differ, for load balancing problem, workload control is. crucial to improve system performance and maintain stability. Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic. Static schemes do not use the system information and are less complex while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic scheme is used here for its flexibility.

DISADVANTAGES

- Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic.
- Static schemes do not use the system information and are less complex.
- A dynamic scheme is used for its flexibility.

III. PROPOSED SYSTEM

Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic. Static schemes do not use the system information and are less complex while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic scheme is used here for its flexibility. The model has a main controller and balancers to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. The system status then provides a basis for choosing the right load balancing strategy. The proposed method will display the pollution rate in area wise .Such things as automobiles, fertilizers, pesticides, energy production (e.g. coal), and agricultural production all contribute pollutants to the surrounding air. Topography can exacerbate the effects of pollutants, trapping them inside a limited area or making it easy for pollutants to settle instead of being swept away by winds.

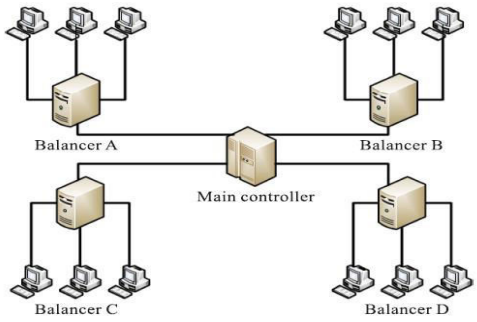


Fig.1. Overall architecture of Proposed System

IV Module Description

System Model

There are several cloud computing categories with this work focused on a public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider. A large public cloud will include many nodes and the nodes in different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations. with the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally. If the cloud partition load status is not normal, this job should be transferred to another partition.

Main controller and balancers:

The load balance solution is done by the main controller and the balancers. The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals with information for each partition, smaller data sets will lead to the higher processing rates. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs.

Assigning jobs to the cloud partition

When a job arrives at the public cloud, the first step is to choose the right partition. The cloud partition status can be divided into three types:

i. Idle: When the percentage of idle nodes exceeds balancer A, change to idle status.
ii. Normal: When the percentage of the normal nodes exceeds balancer B, change to normal load status.
iii. Overload: When the percentage of the overloaded nodes exceeds balancer C, change to overloaded status.

Motivation

A relatively simple method can be used for the partition idle state with a more complex method for the normal state. The load balancers then switch methods as the status changes. Here, the idle status uses an improved Round Robin algorithm while the normal status uses a game theory based load balancing strategy.

Load balance strategy for the idle status

There are many simple load balance algorithm methods such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin. The Round Robin algorithm is used here for its simplicity. The Round Robin algorithm is one of the simplest load balancing algorithms, which passes each new request to the next server in the queue. The algorithm does not record the status of each connection so it has no status information. In the regular Round Robin algorithm, every node has an equal opportunity to be chosen. However, in a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes. Thus, an improved Round Robin algorithm is used, which called "Round Robin based on the load degree evaluation". The algorithm is still fairly simple. Before the Round Robin step, the nodes in the load balancing table are ordered based on the load degree from the lowest to the highest.

Load balancing strategy for the normal status

When the cloud partition is normal, jobs are arriving much faster than in the idle state and the situation is far more complex, so a different strategy is used for the load balancing. Each user wants his jobs completed in the shortest time, so the public cloud needs a method that can complete the jobs of all users with reasonable response time. In proposed a static load balancing strategy based on game theory for distributed systems. And this work provides us with a new review of the load balance problem in the cloud environment. As an implementation of distributed system, the load balancing in the cloud computing environment can be viewed as a game.

V .RESULTS

Fig.2. MAIN PAGE

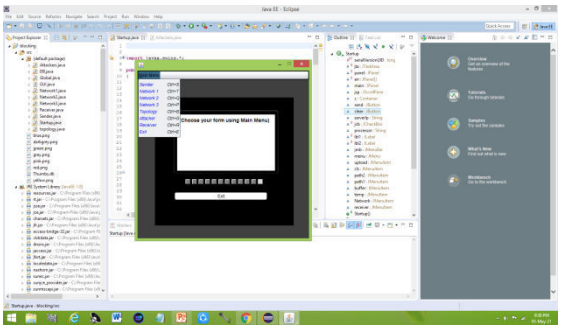


Fig.3. SENDER

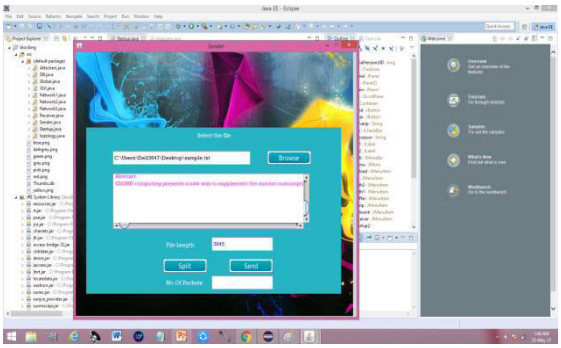


Fig.4. NODE 1

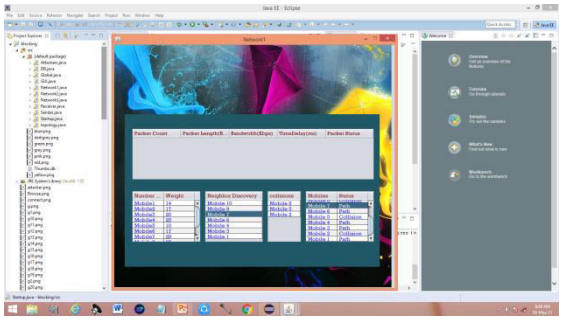


Fig.5. NODE 2

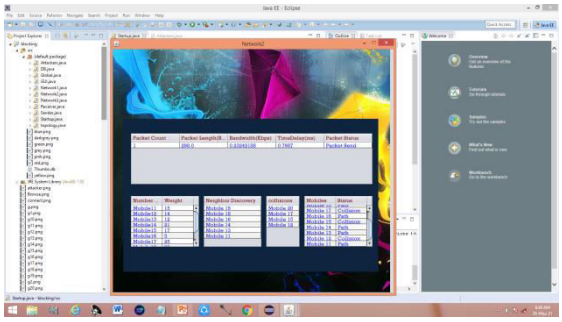


Fig.6. NODE 3

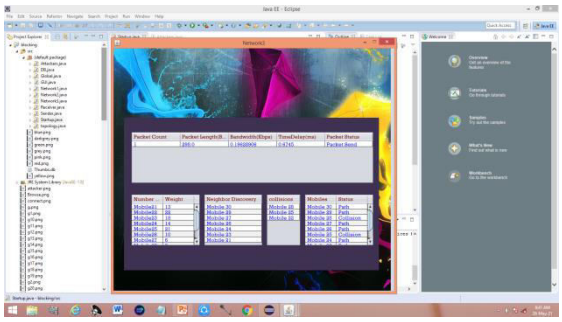


Fig.7. ATTACKER

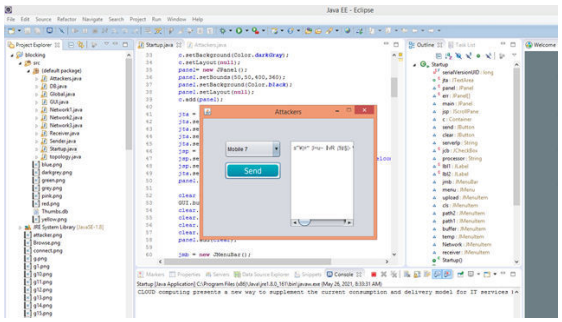


Fig.8. TOPOLOGY

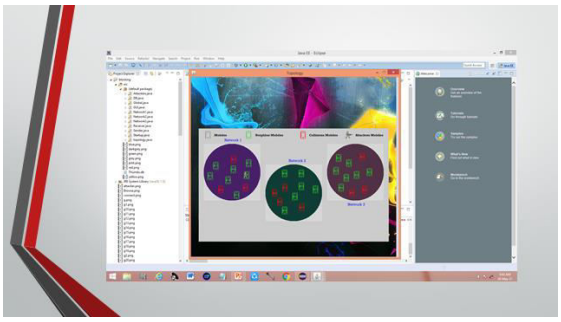
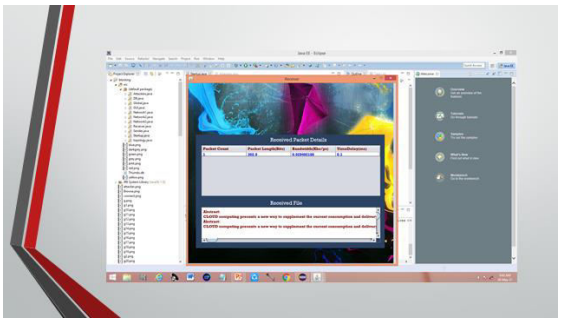


Fig.9. RECIVER



VI. CONCLUSION

We present the following conclusions. Firstly, work sharing among an autonomous local mobile device crowd is a viable method to achieve speedups and save energy. The addition of new resources up to an optimal amount can yield increased speedups and power savings. Secondly, a generalized framework can be used for abstracting methods and enabling parameterization for different types of tasks made of independent jobs. Thirdly, inherent challenges of mobile computing such as random disconnections, having no prior information on participating nodes, and frequent fluctuations in resource availability can be successfully accommodated via fault tolerance methods and work stealing mechanisms.

REFERENCES

1. R. Hunter, The why of cloud, http://www.gartner.com/

DisplayDocument?doc cd=226469&ref= g noreg, 2012.

2. M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, Cloud computing: Distributed internet computing for IT and scientific research, Internet Computing, vol.13, no.5, pp.10-13, Sept.-Oct. 2009.

3. P. Mell and T. Grance, The NIST definition of cloud computing, http://csrc.nist.gov/ publications/nistpubs/800- 145/SP800-145.pdf, 2012.

4. Microsoft Academic Research, Cloud computing, http://libra.msra.cn/Keyword/6051/cloud-computing?query= cloud%20computing, 2012.

5. Google Trends, Cloud computing, http://www.google. com/trends/explore#q=cloud%20computing, 2012.

6. N. G. Shivaratri, P. Krueger, and M. Singhal, Load distributing for locally distributed systems, Computer, vol. 25, no. 12, pp. 33-44, Dec. 1992.

7. B. Adler, Load balancing in the cloud: Tools, tips and techniques, http://www.rightscale. com/info center/whitepapers/Load-Balancing-in-the-Cloud.pdf, 2012

8. Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, Availability and load balancing in cloud computing, presented at the 2011 International Conference on Computer and Software Modeling, Singapore, 2011.

9. K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, N. Nitin, and R. Rastogi, Load balancing of nodes in cloud using ant colony optimization, in Proc. 14th International Conference on Computer Modelling and Simulation (UKSim), Cambridgeshire, United Kingdom, Mar. 2012, pp. 28-30.

10. M. Randles, D. Lamb, and A. Taleb-Bendiab, A comparative study into distributed load balancing algorithms for cloud computing, in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications, Perth, Australia, 2010, pp. 551-556.