iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VI **Month of publication:** June 2022

DOI: https://doi.org/10.22214/ijraset.2022.44975

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

Compute With Nearby Mobile Devices Using Work Stealing Algorithm – Cloud Technology

M. Rachana[1], K. Supraja[2], P. Siri[3], Mr. L. Manikandan[4]
[1, 2, 3]*Undergraduate Student, Department of Computer Science Engineering, Sridevi Women's Engineering College, Hyderabad*
[4]*Assistant Professor, Department of Computer Science Engineering, Sridevi Women's Engineering College, Hyderabad, Telangana*

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 utilized 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 [3].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.***

I. INTRODUCTION

A. Purpose

A different unique model is desired to build and can be differentiated in terms of using only local mobile resources opportunistically, satisfying the requirements of a mobile device cloud of being proactive, opportunistic and load-balanced while showing speedups and energy savings in an actual implementation. Our focus is on a model that can be used to implement a variety of tasks, not limited to query processing, sensing, or human validation.

B. Scope

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, thus creating a resource-saturated ecosystem of machine and human intelligence. However, beyond some traditional web-based applications, 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, not just by creating crowd-sourced computing opportunities needing a human element, but also by solving the resource limitation problem inherent to mobile devices. While there are research projects in areas such as mobile grid computing where mobile work sharing is centrally coordinated by a remote server (HTC power to give1) and crowd-powered systems using mobile devices (Kamino2 , Parko3) a gap exists for supporting collective resource sharing without relying on a remote entity for connectivity and coordination[8].

C. Model Diagram/Overview

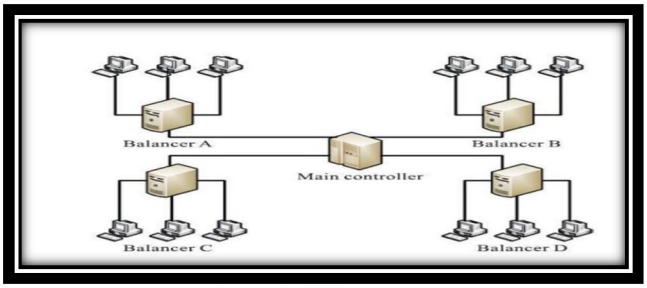


Fig. Model Diagram

The above model diagram depicts the information how Main Device control different mobile device. Here, Main controlling device is termed as Main controller and other devices that are controlled are termed as Balancers.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

II. SYSTEM ANALYSIS

A. *Existing System*

There are several unique features that differentiate mobile crowed 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. Hence, solution from grid/distributed computing cannot be used as they are, and need to be adapted to suit the requirements of mobile crowd environments. We build on previous work where we initially investigated static job farming among a heterogeneous group of mobile devices in, which was followed by a more self adaptive approach in using the „work stealing" method and in where three different mobile crowd sourcing applications were implemented and evaluated. The progress of our research on work share for mobile edge-clouds .

1) *Disadvantages Of Existing System:* Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic. ϖ Static schemes do not use the system information and are less complex. ϖ A dynamic scheme is used for its flexibility

B. *Problem Statement*

Node heterogeneity, unknown worker capability, and dynamism are identified as essential challenges to be addressed when scheduling work among nearby mobile devices. 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

C. *Proposed System*

Our results show that the work stealing methods 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.

In the future we hope to build on this result (social aware task sharing) as an incentive for participation. In, Human expertise is used to answer queries that prove too complicated for search engine and database systems, and in Crowed –search, Image Search on mobile devices is performed with human validation via amazon e-mechanical Truk. A generic spatial crowd sourcing platform using smart phones is discussed in, where queries are based on location information. Mobile Phones are used to collect sensor data on Medusa, according to sensing task specified by users. In Rankr, an online mobile service is used to ask users to rank ideas and photos.

These are primarily concerned with the crowd sourcing aspect, using mobile devices as tools access an online crowd sourcing service that is hosted on a remote server.

In contrast, Honeybee defines the crowd as the surrounding mobiles 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.

1) *Advantages Of Proposed System:* 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 methods, 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[1]. 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

III. SYSTEM REQUIREMENT SPECIFICATION

A. *Functional Requirements*

1) Generating Mobile Devices
2) Preprocess Image Data
3) Local Run Task
4) Offload Task
5) Extension compress and offload Task
6) Response Time
7) Task Execution Time Graph

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

B. Non Functional Requirements

Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000.

1) Data Integrity
2) Scalability
3) Interoperability
4) Reliability

C. Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

1) Operating system : windows, linux
2) Processor : minimum intel i3
3) Ram : minimum 4 GB
4) Hard disk : minimum 250gb

D. Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

1) Python ide 3.7 version
2) Anaconda
3) Jupiter
4) Google colab

IV. SYSTEM DESIGN

A. System Architecture

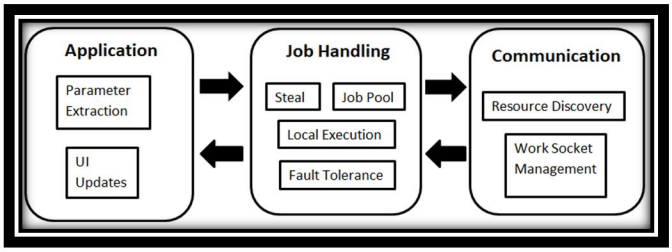


Fig. System Architecture

Honeybee is implemented on Android, using Wi-Fi Direct as the communication protocol [6]. Application developers can use the methods and interfaces provided by the framework for writing work sharing mobile apps.

As shown in Figure, the framework contains three main components responsible for the main areas of Application interfacing, Job Handling, and Communication[9].

1) *The Application Component:* The Application layer method interfaces between application specific code and the core structure. At the starting point of execution, the framework extracts the application specific parameters via the AppRequest interface that provides abstractions to represent the task as a list of jobs.
2) *The Job Handling Component:* The initialization of the job pool begins with processing the AppRequest object. Once the AppRequest task is broken into sub tasks, or jobs, they are stored in two data structures: in an array all Jobs that is not modified throughout execution and in a ConcurrentLinkedQueue jobList, which is subject to change dynamically, as jobs 11 are removed and added from accessing threads[5].

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The array is maintained for validation purposes. During the lifetime of the program, jobList is accessed by several processes as given below:

a) Delegator‟s local job execution thread: Jobs are consumed by polling the jobList from the head.

b) Steal request threads from workers: as steal requests arrive from the Communication component, they are processed by the Job Handling component which creates Callable instances to consume jobs in k chunks from jobList. More than one steal requests may arrive and be processed at the same time.

c) Threads carrying stolen jobs from workers: When the delegator steals jobs from workers, they are passed from the Communication component to the Job handling component, where the job parameters are decoded and assembled into jobs, and added to the jobList.

d) Fault tolerance threads: as fault tolerance mechanisms, jobs that were already assigned to workers may presumed „lost‟ and be added back to jobList. In cases (3) and (4) above, when jobs are added, new local execution threads are spawned as Runnable tasks and added to a single thread pool, thereby ensuring that only one local execution thread is running at a given time. These are then executed locally as described in case (1).

3) The Communication Component

Potential workers are identified by running the resource discovery every t seconds.

.

Table: Types of I/O messages handled by Model

READ	WRITE
1. Steal requests by workers	1. Jobs stolen successfully by workers
2.Workers' acknowledgement of receiving job data	2. Reply to unsuccessful steal attempts by workers (when the delegator does not have any jobs for workers to steal)
3. Negative replies to steal attempts by the delegator (when a worker does not have any jobs)	3. Steal requests from delegator (when the delegator attempts to steal from others)
4. Stolen jobs in cases of successful steals	4. Termination signal sent to workers once delegator verifies all jobs have been completed.
5. Results sent by workers	
6. Worker heartbeats	

Whenever a new resource is detected, the user has the choice to initiate a connection. For each successful worker connection, a reading thread is kept alive throughout the lifetime of the connection as the delegator needs to receive various messages from the workers at intermittent intervals. The messages expected to be received and written by the delegator are summarized in below table.

B. System Components (Modules)

In propose paper author is performing following steps:

1) Module 1 – Local Run Task : Task is done in the local system using this module and this returns required time to execute

2) Module 2 – Generating Mobile Devices: This module deals with generating nearby mobile devices to the inputted mobile device id.

3) Module 3 – Offload Task : Using this module mobile devices are connected using cloud and work is done using two or more mobile devices using work stealing and honeybee algorithm and this returns execution time[7].

4) Module 4 – Extension Offload Task : Using this module, devices are connected in cloud, using pill module offload task is being performed and this returns response time.

5) Module 5 – Task Execution Time Graph : Compares the local run task time and offload task time and Extension Offload task time required to execute and plots a bar graph.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

V. CONCLUSION

The following conclusions can be presented:

1) 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.

2) Secondly, a generalized framework can be used for abstracting methods and enabling parameterization for different types of tasks made of independent jobs.

3) 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.

We report speedups of up to 4 with seven devices and energy savings up to 71% with eight device.

VI. FUTURE ENHANCEMENT

We aim to extend our evaluations to focus on this aspect, using additional criteria such as accuracy and usability in our future work. Further- more, as observed in our experiments, the performance gain plateaus as the number of worker nodes increase due to the additional costs that occur when a single device (delegator as P2P group owner) maintains multiple connections.

To overcome this and scale up, we plan to extend Honeybee to support other topologies and initial experiments in , where an early version of the Honeybee model was ex- tended to support hierarchical Bluetooth connections, show consistent speedups using a linear topology, with an inter- mediate node functioning both as a worker and a delegator.

For this approach, a combination of Bluetooth and Wi-Fi Direct in alternate hierarchical layers can be explored as Wi-Fi direct does not support multiple Wi-Fi direct groups[4].

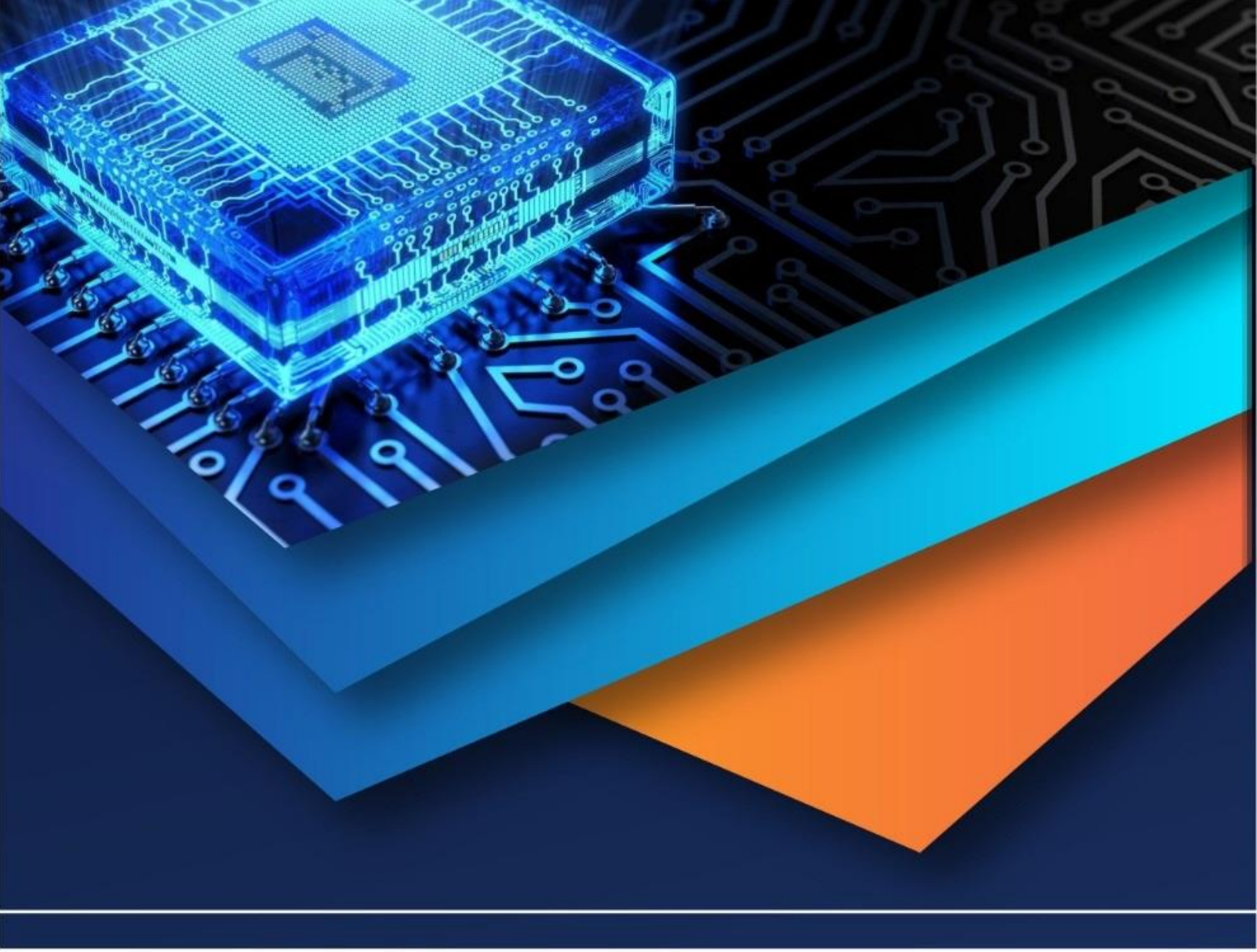
We also plan to experiment with latest D2D technologies such as LTE-Direct to improve performance. In addition, the experiments in this paper were performed in a controlled setting. We plan to extend these tests to more realistic scenarios by using mobility patterns to simulate churn.

REFERENCES

[1] Cisco visual networking index: Global mobile data traffic forecast update. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-nextgeneration-network/white paper c11-520862.html
[2] The hyrax project. http://hyrax.dcc.fc.up.pt/.
[3] K. Agrawal, C.E. Leiserson, and J. Sukha. Executing task graphs using workstealing. In Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on, pages 1–12, April 2010
[4] Bluetooth. Specification of the bluetooth system version 4.1. https://www.bluetooth.org/DocMan/handlers/DownloadDoc. ashx?doc id=282159, December 2013. Accessed: 25/06/2014.
[5] R. Blumofe and C. Leiserson. Scheduling multithreaded computations by work stealing. J. ACM, 46(5):720–748, 1999.
[6] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. Device-to- device communications with wi-fi direct: overview and experi- mentation. Wireless Communications, IEEE, 20(3):96–104, June 2013.
[7] 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.
[8] L. Deboosere, P. Simoens, J. De Wachter, B. Vankeirsbilck, F. De Turck, B. Dhoedt, and P. Demeester. Grid design for mobile thin client computing. Future Generation Computer Systems, 27(6):681 – 693, 2011.
[9] 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.

Textbooks
[1] Programming Python, Mark Lutz
[2] Head First Python, Paul Barry
[3] Core Python Programming,R. Nageswara Rao
[4] Learning with Python, Allen B. Downey

WEBSITES:
[1] https://www.w3schools.com/python/
[2] https://www.tutorialspoint.com/python/index.htm/
[3] https://www.javatpoint.com/python-tutorial
[4] https://www.leanpython.org/
[5] https://www.pythontutorial.net/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)