www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 8 Aug 2015, Page No. 13977-13980

Implementation of mobile crowd network for equalized distribution of work processing

Vandana Sahu[#1], Prof. Santosh Tamboli[*2]

Department of Information technology
VIDYALANKAR INSITUTE OF TECHNOLOGY, MUMBAI UNIVERSITY
Vandana.sahu@vit.edu.in
santosh.tamboli@vit.edu.in

ABSTRACT

In this paper we are going to implement distributed framework for performing a simple computation, on mobile computing devices. A central server should take a large computation and decompose it. Mobile computing devices will then be allowed to connect to the server. Connecting, the devices will assign tasks to complete, and upon completing the tasks, the results will be sent back to the server to be mapped.
The inherent problems of mobile computing such as resource scarcity, security and low connectivity pose problems for most applications.
However, the dynamic nature of mobile computing makes sharing and coordinating work difficult.
We help by pooling together the processing power of mobile devices within a crowd to form mobile cloud. We explore this concept of 'work stealing' crowd computing in a distributed processing on an opportunistic network and focus on the optimized processing of work. Current work stealing mechanism, security can be specified wherein the data transmitted across the crowd network will be secured using encryption techniques and the system Algorithm is generated which states the techniques and working.

Keywords*: Crowd computing, work distribution, Mobile computing.*

I. INTRODUCTION

Crowd computing is a term that has been used only recently in the literature, and has been conceptualized in various ways as being related to crowd sourcing, human computation, social computing, mobile computing

As such different application is in process in different domain like distributed Processing, Vehicular networking and Congestion mitigation. This literature lacks a common definition of crowd computing, and emerging technologies with somewhat similar uses have added to the conceptual confusion.

Some authors have referenced work by other scholars and a few have attempted to provide a definition of crowd computing, it appears that the multiple streams of research and definitions have evolved somewhat independently

This project focus on Mobile crowd computing within the extant literature, in order to propose a definition of crowd Computing in terms of optimization of the work and security focus in term of encryption which can be used to Position the research already conducted on this subject, and can be used as a starting point for further research.

In this project we focus to make Distributed work scheduling to worker and see the optimization of the work of each worker Work stealing has proven to be an effective method for scheduling. Fine-grained parallel programs on multi-core computers. To achieve high performance [1]

II. BACKGROUND OVERVIEW

A. *Existing System*

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 Wi-Fi or 3G is not feasible because of bandwidth issues, data access fees, and the battery drain. Increasing usage and capabilities of smart phones, combined with the potential of crowd computing can provide a collaborative opportunistic resource pool to solve these problems these distribution work stealing focus on the security in that encryption techniques.[2]

Crowd computing has been described in various ways including distribution of human intelligence tasks to mobile devices, cloud computing with humans, human problem solving with large numbers of people using computers, and broadly as a set of human interaction tools for idea exchange and non-hierarchical decision making. From the literature four common characteristics can be identified to define the boundaries of the term, i.e.: participation by a crowd of humans, interaction with computing technology, activity that is predetermined by the initiator or application itself and the execution of tasks by the crowd utilising innate human capabilities[3].

Volunteer computing is a paradigm in which large Numbers of computers, volunteered by members of the general public, provide computing and storage resources. Early volunteer computing projects include the Great Internet Prime Search.

DOI: 10.18535/ijecs/v4i8.62

BOINC (Berkeleyopen infrastructure for Network Computing) is a middleware system for volunteer computing.
Boing is being used by a number of projects, including SETI@home , and others.

Volunteers participate by running a BOINC client program on there computers.
BOINC-based projects are autonomous. Each project operates a server consisting of several components.

WebInterface for account and team management, message board.
Task server that creates task, dispatches them to client
Data server that downloads input files and Execute and upload.

III. Proposed System

We are focusing on the work distribution of the crowd Network and going to implement distributed framework for performing computation. We are utilizing the work distribution concept and can analysis the time and management

Decomposition Techniques

In LU Decomposition, the matrix is decomposed in iterations (See Appendix for description of LU Decomposition) within each iteration, the pivot row (at iteration i, this represents row i) remains constant, while the calculation of all subsequent rows is dependent on the pivot row, and the pivot column. As such, a natural parallelisation scheme is to calculate all rows in parallel. Further parallelisation may be achieved, by parallelising not only the rows, but the columns in each row as well, however this level of granularity was detrimental.

As described in class, the communication time may be modelled as:
Communication Time = T o + n/B
Where,
T o = the time to send an empty message.
n = the message size
B = the link bandwidth

In distributed computing, T is relatively large, while B is relatively small. As such, it is desirable to reduce
the number of messages required. Which in this case, requires increasing the message size, and not using the full Parallelization scheme.
Thus, it was decided that each row in the matrix, be a single task. All processes must wait for all rows to be
Computed, before continuing to the next iteration

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

LU Decomposition:

Lower Upper (LU) decomposition is a form of Gaussian elimination that factors a square matrix as the product of an upper triangular matrix and a lower triangular matrix.

Forward elimination will be used to find the Lower and Upper values of the matrix.
For the Lower:

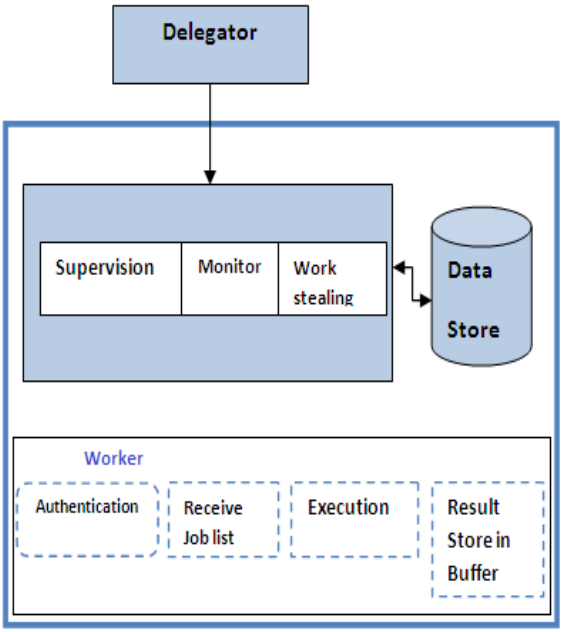
1 11 = a 11/ a11= 1

1 21 = a 21 /a 11
1 31 = a 31 /a 11
...
For the Upper for the first iterations (to reduce first columns 0)
U= A
U 2 = u 2 − u 1 ∗ l 21
u 3 = u 3 − u 1 ∗ l 31
For the next iterations:
U 3 = u 3 − u 2 ∗ l 32
....
Eventually we lead to

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}.$$

IV. System Architecture

The concept of Work Stealing on multi processors was first introduced. 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.



DOI: 10.18535/ijecs/v4i8.62

V. MODULES

Delegator Module

1) *Supervision Module*

In this module the delegator will assign the work equally from his input job list to his assigned workers group.

2) *Monitoring Module*

In this module delegator will monitor the workers and if worker device want to steal the job after completion of his job then at that time delegator will be the 'victim' and the worker will be 'Steal'.

3) *Work Stealing Module*

If delegators own job list is empty, assume the role of 'thief', select a worker device and try to steal jobs. If stealing attempt is successful, run acquired jobs. If not, select a different worker.

Workers Module

1) *Login & Registration*

This is the authentication module of the system facilitating users to add themselves to the system as well as authenticate and utilize the system, thereby providing access to valid registered users in the system.

2) Receiving and Working on Job List

Connect to a delegating device, and receive job list and Start executing the job list. Store the results in the completed buffer.

VI. Algorithm and techniques

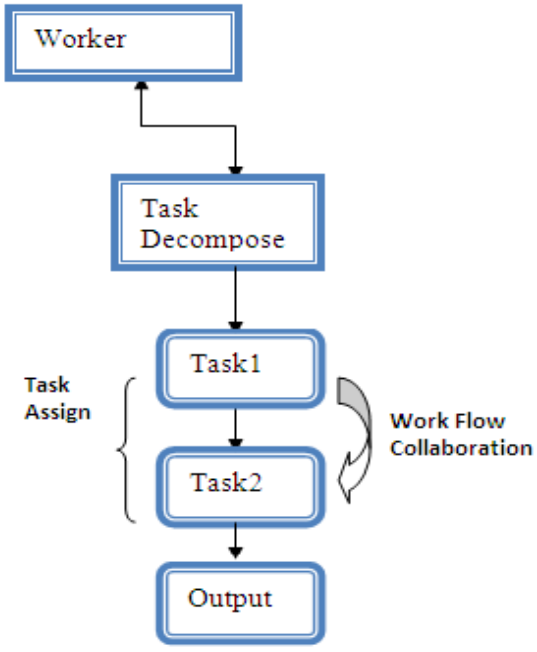
Input: A non-empty list of job parameters. This shall be referred to as the 'Job list'.
Output: An array of computed results corresponding to each job.
In the device where the jobs originate from (delegator)

- Construct job list, and connect to workers.
- Distribute jobs equally among node in secure format
- Start executing the remaining jobs on the list, while
- listening for incoming result from worker
- If a worker device signals that it want to steal examine stealing condition. If met, assume the role of 'victim' and let the worker 'steal'.
- If own job list is empty, assume the role of 'thief', select a worker device and try to steal jobs. If stealing attempt is successful, run acquired jobs. If not select a different worker
- Continue this process until the conditions of job completion are met. Upon completion, signal to all workers that the job has been completed and terminate.

In a worker device,

1) Connect to a delegating device, and receive job list in secure format.
2) Start executing the job list. Store the results in the completed buffer.
3) If the completed buffer is full, transmit the completed list to the delegator.
4) If the delegator signals it want to steal, examine the stealing condition. If met assume role of 'victim'.

Work Process of the system.



This framework is to envision a future of crowd work that can support more complex, creative, and highly valued work. At the highest level, a platform is needed for managing pools of tasks and workers. Complex tasks must be decomposed into smaller subtasks, each designed with particular needs and characteristics which must be assigned to appropriate groups of workers who themselves must be properly motivated, selected, and organized Tasks may be structured through multi-stage workflows in which workers may collaborate either

synchronously or asynchronously. Finally, quality assurance is needed to ensure each worker's output is of high quality and fits together.

CONCLUSION & FUTURE SCOPE

We are focusing on the work distribution of the crowd Network. In our future work, we hope to address security guarantee and integrity. 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 in it.

We are going to implement distributed framework for performing computation, on Mobile computing devices. Central server should take a large computation and decompose it. Mobile computing device will then be allowed to connect to the server. Upon connecting the devices will be assigned tasks to complete, and upon completing the task, the result will be sent back to the server to be mapped. Here are we are utilizing the work distribution concept and can analysis the time and management.

REFERENCES

[1] "Scheduling Parallel Programs by Work Stealing with Private deques", IEEE Transactions On Information Forensics and Security, Vol. 9, No. 4, April 2014

[2] "Mobile Crowd Computing with Work Stealing", IEEE Transactions on network based information system Vol.15, 2012

[3] Crowd computing: a literature review- Kalpana Parshotam (University of Witwatersrand, Johannesburg, South Africa)

[4] High-Performance Task Distribution for Volunteer Computing- space Sciences Laboratory-University of California, Berkeley

[5]http://www.researchgate.net/publication/234113867_Crowd_computing_a_survey

[6]"Adaptive Work Stealing with Parallelism Feedback" Computer Science and Artificial Intelligence Laboratory Massachusetts Institute of Technology Cambridge, MA 02139

[7]" Honeybee: A Programming Framework for Mobile Crowd Computing" Niroshinie Fernando, Seng W Loke, Wenny Rahayu

[8] The case for crowd computing –computer laboratory Cambridge united- kingdom.