Mathematical Theory and Modeling
ISSN 2224-5804 (Paper) ISSN 2225-0522 (Online)
Vol.5, No.7, 2015
www.iiste.org

The Task Allocation Model In Communication Channel Delay

Kamini Raikavar[1*], Virendra Upadhyay[*], Manoj Shukla[**]

[*]Department of Mathematics, MGCGV, Chitrakoot, Govt. Model Science College (autonomous), Jabalpur(M.P.) India

ABSTRACT
In this paper a heuristics approach for task allocation in a distributed computing system has been discussed. This performs static allocation and provide near optimal results. The suggested algorithm is coded in Mat Lab and implemented on a Dual Core machine and found the performance of the developed algorithm is satisfactory.

1. INTRODUCTION

A Distributed Computing System (DCS) consists of any number of possible configurations, such as mainframes, personal computers, workstations, minicomputers, and so on. The goal of distributed computing is to the transparent data distribution within a local network. A user-oriented definition of distributed computing is"Multiple Computers, utilized cooperatively to solve problems" has been reported by {Sita[04],Chia [03], and Bhut [02]}. Distributed computing systems are of current interest for the researchersdue to the advancement of microprocessor technology and computer networks {Till [05], Aror [01]}.In a DCS, the execution of a program may be distributed among several processingelements to reduce the overall cost of execution by taking advantage of inhomogeneous computational capabilities and other resources within the system. The task allocation in a DCS finds extensive applications in the faculties, where large amount of data is to be processed in relatively short period of time, or where real-time computations are required such as, Meteorology, Cryptography, Image Analysis, Signal Processing, Solar and Radar Surveillance, Simulation of VLSI circuits and Industrial process monitoring are areas of such applications.

2. PROBLEM STATEMENT

Consider a DCS consisting a set of n processors P = {p1,p2,….pn}, interconnected by communication links and a set of m tasks T = {t1,t2,…..tm} where m > n. An allocation of tasks to processors is defined by a function Aalloc from the set T of tasks to the set P of processors such that: Aalloc: T→P ,where Aalloc (i) = j if task ti is assigned to processor pj , $1 \leq I \leq m$, $1 \leq j \leq n$.

While designing the model it is assumed that Execution Cost (EC) of each task on all the

processors and the Data Transfer Rate (DTR) between the tasks is known and will be taken in the

form Execution Cost Matrix [ECM (,)] of order m x n, and Data Transfer Rate Matrix [DTRM (,)] of order m respectively. The communication channel delay is also considered and taken in the form of Channel Delay Matrix [CDM (,)] of order m.

3. PROPOSED METHOD

The allocation of tasks is to be carried out so that each task is assigned to a processor whose capabilities are most appropriate for the tasks, and their processing cost is to be minimized. The present model passes through the following phases.

Phase –I

Average Load:

The average load Lavg and Total Load (TL) is to be assigned on processor pj with 05% Tolerance Factor (TF) has to be calculated as:

$$L_{avg}(p_j) = \frac{ec_{ij}}{m} 1 < i < m, 1 < j < n \text{.....3.1}$$

$$TL = \sum L_{avg}(p_j) + TF \text{.....3.2}$$

$$where\ TF = \left(\sum L_{avg}(p_j)\right) * \frac{05}{100}\ and\ \ j = 1,2,...n$$

Selection of “n” Task on the basis of minimum DTR:

The upper diagonal values of the DTRM (,) are stored in Maximum Data Transferred Rate Matrix MAXDTRM (,) of order m(m-1)/2 by 3 the first column represents first task (say rth task), second column represent the second task (say sth task) and third column represent the DTR (drs) between these rth and sth tasks. The MAXDTRM (,) is then stored in ascending order assuming the third column as sorted key and select first tasks “n” which and store the tasks in Ttasks(j) (where j = 1,2,…,n) and also store the remaining m-n tasks in another linear array TNtasks(k) (where k = 1,2,…,m-n).

Assignment: To get the initial assignment store the ECM (,) in NECM (,) NECM (,) ← ECM (,) then reduce the NECM(i,j) (where i = 1,2,…n and j = 1,2,…n) in the square matrix of order n by deleting tasks stored in TNtasks(k) which is the intersection of Ttasks(i) and apply the YAS-Algorithm developed by Yadav et al [04]. The initial allocation is stored in an array Tass(j) (where j = 1,2,…,n) and also the processor position are stored in a another linear array

Aalloc (j). Get the value of TTASK (j) by adding the values of Aalloc (j) if a task is assigned to a processor otherwise continue. Select a task from TNtasks (k) (where k = 1,2,……m-n) for assignment say tk and assign task to processor pj where the value of EC in minimum. Store the assignment and their position in Tass(j) and Aalloc (j) respectively and modify the TTASK (j) by adding the value of Aalloc (j). The TNtasks (k) is also modify the by deleting the tasks tk. This process of assignment is continuing till the remaining "m-n" tasks are get allocated.

Phase -II

After completing the allocations the Processor's Wise Execution Cost (PEC) ecij ($1 \le i \le m$, $1 \le j \le n$) of each task ti on the processor pj is calculated using the following equation

$$PEC(A_{alloc})_j = \sum_{\substack{1 \le i \le m \\ i \in TS_j}} ec_i, A_{alloc(j)} \text{.....3.3}$$

$$where TS_j = \{i : A_{alloc}(i) = j, j = 1,2...n$$

Inter Processor Communication (IPC):

The Inter Processor Communication cost of the interacting tasks ti and tk is depends upon the per

unit data transfer rate dik during the program execution is determined by using the equation given below Inter Processor Communication with delay (IPCWD) is calculated by the equation given below

IPCWD $(A_{alloc})_j = \left[\min(ec_{ij} * d_{ij})\right] * cd_{ij}$, j = 1,2,…..n and i =.1,2….m ……3.4

Where cd_{ij} is the delay channel.

Inter Processor Communication without delay (IPCWOD) is calculated by the equation given below

IPCWOD $(A_{alloc})_j = \left[\min(ec_{ij} * d_{ij})\right]$ j = 1,2,…..n and i =.1,2….m ……..3.4.1

Calculate the Overall Processors Cost [OPC] with delay and without delay calculated as

OPCWD $(A_{alloc})_j = PEC(A_{alloc})_j$ + IPCWD $(A_{alloc})_j$ ……3.5

OPCWOD $(A_{alloc})_j = PEC(A_{alloc})_j$ + IPCWOD $(A_{alloc})_j$ ……3.5.1

and find the average Overall Processors Cost [OPC]

The Mean Service Rate [MSR] with delay and without delay of the processors are calculated by using the equation described below and store the results in MSR $(A_{alloc})_j$ (where j = 1,2,…,n).

$$MSRWD(A_{alloc})_j = \frac{1}{\text{OPCWD}(\text{A}_{\text{alloc}})_{\text{j}}} where \quad j = 1,2....n \qquad \text{……..3.6}$$

$$MSRWD(A_{alloc})_j = \frac{1}{\text{OPCWD}(\text{A}_{\text{alloc}})_{\text{j}}} where = 1,2....n \qquad \text{.....3.6.1}$$

The throughputs of the processors with delay and without delay are calculated by using the equation given below and store the results in a linear arrays TRP $(A_{alloc})_j$, where j=1,2…….,n.

$$TRP(A_{alloc})_j = \frac{TTASK(A_{alloc})_j}{\text{OPCWD}(\text{A}_{\text{alloc}})_{\text{j}}} where \quad j = 1,2....n \qquad \text{.....} \qquad \text{.....3.7}$$

$$TRPWOD(A_{alloc})_j = \frac{TTASK(A_{alloc})_j}{\text{OPCWOD}(\text{A}_{\text{alloc}})_{\text{j}}} where \quad j = 1,2....n \qquad \text{.....3.7.1}$$

Finally, Critical Transmission Delay [CTD] and the Optimal Processing Cost (OPC) with delay

and without delay have been determined. The maximum value of OPC $(A_{alloc})_j$ will be the Total System Cost (TSC) which shell be the optimal cost of the DCS.

3.4 ALGORITHM:

Step1: Input; m, n, ECM (,), DTRM (,) and CDM (,)

Step2: for i ← 1 to m do

for j ← 1 to n do

Determine the Total Load (TL) to be assigned on processor pj with 05% Tolerance Factor (TF) by using the equation (1) and (2) respectively.

repeat

repeat

Step3: k ← m(m-1)/2

t ← m-n

for i ← 1 to m do

for j← 1 to k do

Store the "k" upper diagonal value of DTRM (,) in MAXDTRM (,)

MAXDTRM (,)← DTRM (,)

Arrange the MAXDTRM (,) ascending order

repeat

repeat

Step4: for i ← 1 to n do

for j← 1 to n do

Store the ECM (,) in NECM (,)

NECM (,)← ECM (,)

repeat

repeat

Step5: count ← n

for i ← 1 to m do

for j ← 1 to k do

Pick-up the minimum value from MAXDTRM (,) say tj , tk

If n = count;

then

store the tasks in a linear array Ttasks( ) and go to the step-5

else ;

repeat for i

if MAXDTRM(i,j)< MAXDTRM(i,k)

bmin ← MAXDTRM(i,j) until j = k

bmin < MAXDTRM(i,k) until j = k

end if

check the corresponding position of bmin MAXDTRM(,) (say tl) and give one increment to count

count ← count+1

store the tasks in a linear array Ttasks( )also store the remaining m-n tasks inTNtasks( )

repeat

repeat

Step 6: Reduce MAXDTRM (,) and also modify the NECM (,) by eliminating the tasks stored in Ttasks( )

Step-7: Apply the YAS-Algorithm to get the initial allocation and store the assignment in an linear array

Tass(j) (where j = 1,2,…,n) also the processor position are stored in a another linear array Aalloc(j). Get the

value of TTASK (j) by adding the values of Aalloc (j) if a task

is assigned to a processor otherwise continue.

Step-7.1:

for i ← 1 to m do

store the TNtasks( ) in Tnon-ass( )

Tnon-ass( ) ←TNtasks( )

repeat

Step:7.2:

for i ← 1 to m-n do

for j ← 1 to n

Select a task from TNtasks (k) (where k = 1,2,…,m-n) for assignment say ti and

assign task to processor pj where the value of ecij is minimum.

Aalloc(k) ← j;

nomade ← nomade + 1;

Tass←Tass ∪ {tk};

repeat

repeat

This process is continuing till the remaining "m-n" tasks are get allocated.

Step-8:

for k ← 1 to m do

for j ← 1 to n do

Compute the final PEC (Aalloc)j using the equation (3.3)

repeat

repeat

Step-9:

for k ← 1 to m do

for j ← 1 to n do

Compute the Inter Processor Communication Cost IPC (Aalloc) j of the interacting tasks ti and tk by using the equation (3.4) and (3.4.1)

repeat

repeat

Step-10:

for i ← 1 to m do

Determine the finally, Overall Processors Cost OPC (Aalloc) j by using the equation (3.5) and (3.5.1)

repeat

Step-11:

for i ← 1 to n do

Compute the mean service rate MSR (Aalloc)j by using the equation(3.6) and (3.6.1)

repeat

Step-12:

for i ← 1 to n do

Compute the processor's throughput by using the equation (3.7) and (3.7.1)

Repeat

Step-13: Calculate the total Critical Transmission Delay [CTD] and the maximum value of OPC (Aalloc)j i.e .the Total System Cost

Step-14:

Stop.

3.5 IMPLEMENTATION OF THE ALGORITHM:

Example:

To justify the application and usefulness of the present algorithm an example of a DCS is considered which is consisting of n = 3 the set of processors P = {p_1, p_2, p_3} connected by an arbitrary network and m = 6 the set of tasks T= {t_1, t_2, t_3, t_4, t_5, t_6} which may be portion of an executable code or a data file.

Step1:

Input of the Algorithm: Data required by the algorithm is given below:
Number of processors available in the system (n) = 3
Number of tasks to be executed (m) = 8

	p_1	p_2	p_3
t_1	6	3	5
t_2	4	2	3
t_3	3	1	2
t_4	5	2	∞
t_5	3	4	2
t_6	6	∞	6
t_7	5	6	7
t_8	∞	2	5

DTRM(,) =

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0.000	0.333	0.250	0.500	0.167	0.125	1.000	0.000
t_2	0.333	0.000	0.000	0.000	0.000	0.000	0.000	0.200
t_3	0.250	0.000	0.000	0.250	0.333	0.500	0.000	0.000
t_4	0.500	0.000	0.250	0.000	0.200	0.333	0.500	0.200
t_5	0.167	0.000	0.333	0.200	0.000	0.000	0.000	0.000
t_6	0.125	0.000	0.500	0.333	0.000	0.000	0.167	0.125
t_7	1.000	0.000	0.000	0.500	0.000	0.167	0.000	0.200
t_8	0.000	0.200	0.000	0.200	0.000	0.125	0.200	0.000

CDM(,) =

	t1	t2	t3	t4	t5	t6	t7	t8
t1	0	2	1	1	2	2	1	1
t2	2	0	1	1	1	1	1	2
t3	1	1	0	2	1	2	1	1
t4	1	1	2	0	1	2	1	2
t5	2	1	1	1	0	1	1	1
t6	2	1	2	2	1	0	2	2
t7	1	1	1	1	1	2	0	2
t8	1	2	1	2	1	2	2	0

Step2:

After Implementation of the Algorithm's steps the Average load to be assigned on the processors

has been given in table 3.1

Table 3.1

Processors	Average execution cost
P1	5
P2	3
P3	4
Total Load	12
05% Tolerance Factor [TF]	0.6 (= 0.5 say)
Total Average load on the processor= Total load + TF]	12+0.5= 12.5

Step3:

The m (m-1)/2 i.e. 28 value of upper diagonal values of the DTRM (,) are stored in MAXDTRM
(,):

MAXDTRM (,) =

t1	t2	0.333
t1	t3	0.250
t1	t4	0.500
t1	t5	0.167
t1	t6	0.125
t1	t7	1.000
t1	t8	0.000
t2	t3	0.000
t2	t4	0.000
t2	t5	0.000
t2	t6	0.000
t2	t7	0.000
t2	t8	0.200
t3	t4	0.250
t3	t5	0.333
t3	t6	0.500
t3	t7	0.000
t3	t8	0.000
t4	t5	0.200
t4	t6	0.333
t4	t7	0.500
t4	t8	0.200
t5	t6	0.000
t5	t7	0.000
t5	t8	0.000
t6	t7	0.176
t6	t8	0.125
t7	t8	0.200

The MAXDTRM (,) is stored in ascending order assuming the third column as sorted key and select those tasks "n=3" which has minimum DTR and store the tasks in Ttasks(j) (where j = 1,2,…,n) and also store the remaining m-n tasks in another linear array TNtasks(k) (where k = 1,2,…,m-n).

MAXDTRM (,) =

t1	t8	0.000
t2	t3	0.000
t2	t4	0.000
t2	t5	0.000
t2	t6	0.000
t2	t7	0.000
t3	t7	0.000
t3	t8	0.000
t5	t6	0.000
t5	t7	0.000
t5	t8	0.000
t1	t6	0.125
t6	t8	0.125
t1	t5	0.167
t6	t7	0.176
t2	t8	0.200
t4	t5	0.200
t4	t8	0.200
t7	t8	0.200
t1	t3	0.250
t3	t4	0.250
t1	t2	0.333
t3	t5	0.333
t4	t6	0.333
t1	t4	0.500
t3	t6	0.500
t4	t7	0.500
t1	t7	1.000

Step4: Store the ECM (,) in NECM,) as:

NECM(,) =

	p1	p2	p3
t1	6	3	5
t2	4	2	3
t3	3	1	2
t4	5	2	∞
t5	3	4	2
t6	6	∞	6
t7	5	6	7
t8	∞	2	5

← ECM(,) =

	p1	p2	p3
t1	6	3	5
t2	4	2	3
t3	3	1	2
t4	5	2	∞
t5	3	4	2
t6	6	∞	6
t7	5	6	7
t8	∞	2	5

Step5:

Ttasks( ) = { t1 t8 t5}

TNtasks( ) ={ t2 t3 t4 t6 t7 }

Step6:

Reduced NECM(,) and MXDTRM(,)

NECM(,) =

	p1	p2	p3
t1	6	3	5
t5	3	4	2
t8	∞	2	5

MAXDTRM (,) =

t2	t3	0.000
t2	t4	0.000
t2	t5	0.000
t2	t6	0.000
t2	t7	0.000
t3	t7	0.000
t3	t8	0.000
t5	t6	0.000
t5	t7	0.000
t1	t6	0.125
t6	t8	0.125
t1	t5	0.167
t6	t7	0.176
t2	t8	0.200
t4	t5	0.200
t4	t8	0.200
t7	t8	0.200
t1	t3	0.250
t3	t4	0.250
t1	t2	0.333
t3	t5	0.333
t4	t6	0.333
t1	t4	0.500
t3	t6	0.500
t4	t7	0.500
t1	t7	1.000

Step7:

Initial assignments are obtained by applying the YAS-Algorithm developed by Yadav et al [Yada04] are given in Table 3.2:

Table 3.2 Initial Assignment

Tasks	Processor	EC
t5	p1	3
t8	p3	5
t1	p2	3

Step7.1:
Tnon-ass () ← TNtasks( )
Tnon-ass () ={ t2 t3 t4 t6 t7 }
Aalloc (j) = {1, 3, 2} and TTASK (j) = { 1,1,1}
After Implementing the Step-7.2 the following final assignment are obtained.
Table 3.3: Final Assignment

Tasks	Processor	EC
t3 + t5 + t7	p1	11
t1 + t2 + t4	p2	07
t6+ t8	p3	11

The final Aalloc(j) = {1,3,2,1,1,2,2,3,3} and TTASK (j) = { 3,3,2}, Table 3.3 shows the final assignment after Implementation of assignment procedure described in section 3.3 in the proposed method. On applying the further steps 8 to 13 following result are obtained and shown
in the table 3.4

Table – 3.4:

Processors	EC	IPC without Delay	IPC with Delay	OPC without Delay	OPC with Delay	MSR without Delay	MSR with Delay	TRP without Delay	TRP with Delay	Total critical Transmission Delay
1	2	3	4	5=2+3	6=2+4	7	8	9	10	11=6-5
P1	11.000	5.079	7.658	16.079	18.658	0.062	0.054	0.124	0.107	2.579
P2	7.000	4.882	7.364	10.882	13.364	0.092	0.075	0.276	0.224	2.482
P3	11.000	3.705	6.510	13.705	16.510	0.073	0.061	0.219	0.182	2.805

3.6 CONCLUSION

Task allocation problem is one of the most important research areas of the distributed computing system. The model discussed in this chapter based on the consideration of execution cost, data transfer rate between the tasks and communication channel delay amongst the tasks.The problem taken into consideration here is of static allocation type problem. The goal ofproposed algorithm is the improving performance of the distributed computer system byminimizing the processing cost of the execution of tasks of a program with and withoutcommunication channel delay.

References

[01] Arora, R.K., and Rana, S.P., "On module assignment in two processors distributed Systems", Information Processing Letters, Vol. 9, No. 3, pp. 113-117, 1979.
[02] Bhutani, K.K., "Distributed Computing", The Indian Journal ofTelecommu nication, pp. 41-44, 1994.
[03] Chiao – Pin Bao, Ming-Chi Tsai, Meei-Ing Tsai., 2007, "A new Approach to Study the Multi Objective Assignment Problem", WHAMPOA- An Interdisciplinary journal, 53 (2007), PP. 123- 132.
[04] Sitaram, B. R., "Distributed computing – a user's view point", CSI Communications, Vol. 18, No. 10, pp.26-28, 1995.
[05] Tillman, F. A., Hwang, C. L. and Kuo, W. (1977), Determining Component Reliability and Redundancy for Optimum System Reliability, IEEE Transactions on Reliability, vol. R-26, 162-165.

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar