International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-11, September 2019

"A hybrid Method to augment the efficiency of Distributed Computing System by DAG -Using Finest Task Allocation with Dual Mode Processors"

Prasant Singh Yadav, Pradeep Kumar Yadav, K.P Yadav, Sunil kumar Bharti

Abstract: ***Distributed computing system creates or provides a platform having multiple computing nodes linked in a specified manner. On the basis of literature review of last few decades it has been noticed that most of distributed computing researchers have shown their effort to maintain load balancing between processors ,effective task scheduling and optimizing different parameters affecting execution cost and throughput .With these above scenario an additional parameter "Self reconfiguration of CPU" is also a countable parameter to augment the efficiency of distributed computing system .Through this research paper we want to present new approach of adaptive scheduling algorithm which is the mix output of effective task allocation to processor involved in computing and self-reconfiguration of those processors as per need of computing. By this proposed method we will optimize the execution cost, service rate and maximize the throughput as an outcome of organized processors consist in heterogeneous distributed computing system, resulting provide the considerable enhancement in the performance of Distributed computing environment.***

Keywords: DCS, Task allocation, CPU Self- Reconfiguration, Task cluster, Execution Cost, PSR, AWS, SAUCLAB, and TPC-w Benchmark.

I. INTRODUCTION:

A system having a bunch of connected processor with high speed network link to execution of parallel tasks. Scheduling a set of task to distributed processors is very critical and important issue. Its solution provides better performance and interaction among nodes. Systematic scheduling in distributed processing environment is a significant issue to advance performance of the system [1, 2]. For this systematically steps are necessary to stop degradation in throughput. Allocation policy has to way one is static and another is dynamic. The problem of finding a best dynamic assignment of a standard program for a two-processor system has been analyzed by [3]. One live utility of a general distributed ADP system is that the system's ability to produce grade of performance proportionate to the degree of multiplicity of resources gift among the system. Taxonomy of approaches to the resource management downside is reportable [4].

Manuscript published on 30 September 2019.
***Correspondence Author(s)**
Prasant Singh Yadav, Ph.D CS Scholar E, Dr APJ AKTU Lucknow India
Dr. Pradeep .Kumar Yadav, principal Scientist CSIR-CBRI Roorkee India
Dr.K.P Yadav, Vice Chancellor, Sangam University, Bhilwara Rajsthan, India
Dr.Sunil Bharti, Professor Galgotia College of Engineering, G Noida,India

The taxonomy, given and mentioned in terms of distributed programming, is additionally applicable to most sorts of resource management. A model for allocating data files has been rumored by [5]. This model considers storage price, transmission price, file lengths, and request rates, likewise as change rates of files, the utmost allowable expected access times to files at each laptop, and therefore the storage capability of every laptop. The model is developed into a nonlinear range zero-one programming downside, which can be reduced to a linear zero-one programming downside. Legendary solutions of AN outsized vary of necessary (and difficult) machine problems remarked as NP-complete issues rely upon enumeration techniques that examine all doable alternatives. The look of enumeration schemes in very distributed surroundings are rumored by [6]. A task allocation model that allocates application tasks among processors in distributed computing systems satisfying: 1) minimum inter-processor communication worth, 2) balanced utilization of each processor, and 3) all engineering application needs has been rumored by Perng-Yi Richard Ma et.al [7]. This downside of task allocation in heterogeneous distributed systems with the goal of increasing the system responsibleness has been addressed [8]. The model relies on the well-known simulated hardening (SA) technique. Yadav et al have rumored AN algorithmic program for responsibleness analysis of distributed system supported failure knowledge analysis [9]. AN economical algorithmic program for best tasks allocation through optimizing responsibleness index in heterogeneous distributed process system has been mentioned by [10]. J. B. Sinclair [11] thought-about the matter of finding a best assignment of the modules of a program to processors in a very distributed system. A module incurs AN execution price which will diverge for every processor assignment, and therefore the modules that don't seem to be assigned to an equivalent processor however communicate with each other incur a communication price. A best resolution to the matter of allocating act periodic tasks to heterogeneous process nodes (PNs) in a very distributed period of time system has been rumored [12]. Matrix reduction technique has been employed by Sagar et al, consistent with the factors given in this a task is chosen at random to begin, with so assigned to a processor [13]. a quick algorithmic program for allocation task in distributed process system has been rumored by Kumar et al [14, 16]. During this technique the author tried to cluster heavily communicated tasks and allotted them to same processor.

AN economical algorithmic program for allocating tasks to processors in a very distributed system has additionally been rumored by Kumar et al. [15].Distributed computing systems provide the potential for improved performance and resource sharing.

To form the foremost effective use of the method power obtainable in multi-processing system, it's essential to assign the tasks dynamically to it processor whose characteristics are most acceptable for the execution of the tasks in distributed process system [17, 18].

The new methodology augments the maximally connected module conception by victimization random techniques and by adding constructs that take into thought the restricted and uneven distribution of hardware resources (often related to heterogeneous systems) has been mentioned by Elsade et al. Authors used pure simulated hardening and also the irregular algorithmic program for randomly-generated systems and artificial structures that were derived from real-world issues [19]. Communication technology has conjointly opened several avenues for process, sharing and transferring the information. Yadav, et al [20] rumored task programming in laptop communication network. a man-made Neural Network (AANN) primarily based task programming model has been mentioned by Yadav et al [21]. For developing the model authors used feedback neural specification. Singh et al reported associate degree ANN based model for Load Distribution in fully connected shopper server network [22].Analysis of load distribution in distributed process systems through systematic allocation task associate degreed an thoroughgoing approach of performance analysis to the distributed systems supported price assignments are rumored by Kumar et al [23, 24]. the needs of task programming in distributed systems square measure maximizing system output by distribution correct tasks to correct processors, maximizing resource utilization, minimizing execution time, minimizing price on the user facet and satisfying economic constraints. [25–27] Distributed systems order acceptable machine power, which may be used as an answer of problems with giant machine necessities. There square measure varied varieties of programming algorithms. The importance of digital computer task programming issues ends up in several comparative studies. Kwok et al. [28] extensively classified, delineated and compared twenty seven static programming algorithms with their functionalities through a nine-task downside. Braun et al. [29] gave a comparison of 11 heuristics for mapping meta-tasks while not communication delays onto a heterogeneous cluster of processors. List programming techniques also are wide utilized in determination task programming problems. List programming techniques assign a priority to tasks that square measure ready to be dead supported a particular heuristic, then kind the list of tasks in decreasing priority. List programming may be a cooperative static technique. Alternative static dependent techniques that we have a tendency to study square measure cluster algorithms also as duplication primarily based algorithms. Davidovic et al. [30] studied the comparison of list programming approaches. Alternative technique that has attracted several researches [31] in parallel computing is Genetic Algorithms (GA), in the main as a result of GA is effective in determination laborious issues.

Our creation is as below:

- We build up a heuristic form for task or segment of task that can be used in parallel
- We define the task arrangement and processor Self-configuration issues here we use Dual mode processor.
- We Define and explain how CPU (Processor) can reconfigure itself in different task allocation condition.
- We develop a heuristic model that is the mix capsule of adaptive task scheduling and adaptive CPU (Processor) Self-Reconfiguration techniques which will optimize the execution cost and maximize the throughput.
- Finally develop self-adaptive technique for Heterogeneous Distributed computing system computing system.

II. NOTATIONS:

N means quantity of processors

M means quantity of task

N is the quantity of processor

N_C is the quantity of cluster

TS is the total service time

IMCC intermodal Communication cost

EC is the communiqué cost

ITCC inter task communication cost

S means service rate

T is the throughput

C_a is Processor capacity

W_L is assigned work load

MXCC is maximum communication cost

2.1 Task allotment problem

Specific task allotment drawback being self-addressed as follows: thought-about Associate in Nursing applications program consisting a group having "m" tasks $T = \{t_1, t_2, t_3 \ldots t_m)$ and a heterogeneous distributed process system consisting a set of "n" processors $P = \{p_1, p_2, p_3 \ldots p_n\}$, here we consider that m > n, and allocated every one of m tasks to one of the n processors in such a style in order to minimize total system time and balancing the processing load to respective processor . While developing the model following inputs have been taken into consideration.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

I. Per Bit Processor Service Rate (PSR),
II. Task Size (TS),
III. Inter Task Communication Cost (ITCC)
IV. Execution Cost (EC).

2.1 Processor's Execution Rate (PER): Per Bit Processor's service rate which is the execution rate e_{rj} $(1 \le j \le n)$ of each processor is the speed (bytes/ second) of the processor at which they execute the tasks given in the form PSR(j) (where j=1,2,3.....,n).

$$PSR(j)= \begin{vmatrix} er1 \\ er2 \\ er3 \\ . \\ . \\ ern \end{vmatrix}$$

Transpose of PSR (j)

$$PSR\ (j)' = \begin{vmatrix} er_1 & er_2 & er_3 & . & . & er_n \end{vmatrix}$$

2.2 Task Size (TS): A task is a sequential program, which performs some predefined action and possibly communicates with other tasks in a system. The task size t_{si} $(1 \le i \le m)$ of each task depends on the length of tasks and generally counted in bytes. Here task size follow linear array format TS (i) (where i= 1, 2, 3.....m)

$$TS(i)= \begin{vmatrix} ts1 \\ ts2 \\ ts3 \\ . \\ . \\ tsm \end{vmatrix}$$

2.3 Inter Task or Inter-module Communication Cost (ITCC): when task t_i and t_k with each other then communiqué cost cc_{ik} is incur because of trade of data units between them, while the execution of respective process. The ITCC is taken in the form of a symmetric matrix named as Inter Task communiqué Cost Matrix (ITCCM), which is of order m.

ITCC(i,k)=

	t1	t2	t3	.	.	tm	
t1	cc11	cc12	cc13	.	.	cc1m	
t2	cc21	cc22	cc23	.	.	cc2m	
t3	cc31	cc32	cc33	.	.	cc3m	
.	.	.	.	.	.	.	
tm	ccm1	ccm2	ccm3	.	.	ccmm	mxm

2.4 Execution Cost (EC): The execution cost er_{ij} Where $1 \le i \le m$, $1 \le j \le n$ processor p_j is responsible to process the assigned task t_i. To determine EC, initially we have taken the product of transpose of the PSR (j) and TS (i) and stored the result in Execution Cost Matrix (ECM(i,j)) of order m x n. After multiplication, the computed final ECM (i, j) is as follow: Where $er_n * ts_m = ec_{ij}$ (i=1, 2…..m and j=1, 2…..n) here Hungarian method is used for allotment of tasks to processors to the set P of processors from defined set of task or cluster.

ECM(i,j)=

	p1	p2	p3	.	Pn	
t1	er1*ts1	er2*ts1	er3*ts1	.	ern*ts1	
t2	er1*ts2	er2*ts2	er3*ts2	.	ern*ts2	
t3	er1*ts3	er2*ts3	er3*ts3	.	ern*ts3	
.	.	.	.	.	.	
.	.	.	.	.	.	
tm	er1*tsm	er2*tsm	er1*tsm	.	ern*tsm	Mxn

III. SUPPOSITIONS TAKEN:

- The state of task is either operational or failed
- There is no preemption of resources, assigned task remains on processor until its execution accomplished.
- Set of task is residing on same processor have zero inter task communiqué cost.
- Different processor, communication path or link produces different cost for same task assigned to them.
- When the processor or system is switch to alternative configuration the performance change quantitatively.
- The servers are designed in such a manner that they can distribute memory and processing resource to logical subsystem.
- The real time Self-reconfiguration processor is used to exploit performance with changing load.
- Self-reconfiguration of system will be in terms of hardware reconfiguration or software reconfiguration.
- Self-reconfiguration by the processor can be adapting on demand from available online such service providing platform.
- Self-Reconfiguration Seek time will be compensate by adopted configuration of respective processor.

IV. CPU (processor) self-Reconfiguration:

Heterogeneous Distributed computing demand ever increasing ability and performance so as to address dynamical user necessities, enhancements insystem options, dynamical protocol and data-coding standards, and demands for support of a range of various user applications several rising applications in communication, computing and client physical science demand that their practicality stays flexible when the system has been factory-made. What is more, these days analysis is pushing forward, searching for complicated heterogeneous, and reconfigurable multi-cores design. Smart samples of heterogeneous systems, extremely dynamic in content, work and infrastructure (i.e., nodes are unendingly exploit and joining) are cloud computing, grid, cluster and peer to look architectures.

Retrieval Number: K19740981119/19©BEIESP
DOI: 10.35940/ijitee.K1974.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

So as to beat the boundaries etymologizing by the increasing complexness and therefore the associated work to take care of such complicated infrastructures, one chance is to adopt self-adaptive and involuntary computing systems. These systems are ready to configure, heal, optimize and shield them while not the necessity for human intervention. inside this context, reconfigurable computing systems [2] are moving to self-adaptive and involuntary computing systems wherever hardware parts, the applications and therefore the software have to be compelled to be seen as associate distinctive entity that require to be able to autonomously adapt itself to comprehend the foremost effective performance. Distributed system will optimize its performance by learning to reconfigure mainframe and memory resources in reaction to current work. We will use a learning framework that uses commonplace system-monitoring tools to spot preferred configurations and their quantitative performance effects [2]. Learning frame works as a self-adaptive and involuntary system that is ready to configure, heal, optimize and defend itself while not the requirement for human intervention. so as to realize such a state of affairs, the self-adaptive and involuntary computing systems need to be able to monitor its behavior to self-update itself, in one, or during a combination of many, of its parts (hardware design, OS and running applications), to beat potential failure in accomplishing its tasks System will improve its performance if it can adapt its configuration because the work changes. As our analysis s associated with high computing computation in heterogeneous distributed system (HDCS), thus such application typically needs high network performance, quick storage, and great deal of memory, high computing capabilities or all of those. we tend to are victimization some on-line cloud mating portals like AWS, Sauce lab, Bootstrap etc. that allows US to running such computing in cloud and scaling to larger no of parallel task than would be sensible in most on premises atmosphere. Such on-line portal facilitate US to cut back value and time by providing mainframe, GPU and FPGA servers on demand, optimized for specific applications for applications with short term, unpredictable which will be uninterrupted, work which will tolerate interruption, versatile begin and finish time and application with steady state usage.

In literature we have found big performance profit to reconfiguration in response to load changes. We can construct an implementation of the on top of benchmark victimization normally offered hardware and software package. These benchmark provides a well-defined simulation of vary the employment and measure the system below varied demands. Findings represent 3 vital steps toward the ultimate goal of constructing a completely accommodative self-reconfiguration system: (a) we tend to establish that dynamically reconfiguring hardware in response to employment changes has the potential to enhance performance. (b).Using un-instrumented middleware (c). Provide solely raw and low-level system statistics. Thus it's attainable to predict that configurations can surpass the opposite at any given time. Time period reconfiguration may be required to maximize performance below a variable employment. we can adapt any required configuration on demand online from such service provider .Obtained reconfiguration will also compensate the time taken in adapting configuration in hardware or software term.

V. PROPOSED METHOD

We planned a heuristic tasks allocation model is proposed for resolution a load leveling tasks allocation drawback. Here we present new approach of adaptive scheduling algorithm which is the mix output of effective task allocation to dual mode processor which can work in Normal mode and self-reconfiguration mode (processors can configure themselves as per need of computing with varying load).

Step 1: Inputs:

(a): A program having m tasks{ t1, t2….tm}.

(b): A set P = {p1, p2….pn} of n dual mode processors.

Step 2: Task choice order: Since the ranges of the tasks are quite number of processors, thus the priority for his or her execution to be set supported their execution and communication prices. The tasks choice list, Tnon-asg is generated by sorting the tasks with reference to increasing order of their value perform CF(ti) of execution of i^{th} task on j^{th} processor having communication cost C are calculated as:

$$\underset{1\leq i\approx m}{CF(t_i)} = \frac{\sum_{j=1}^{n} e_{ij}}{n} + \underset{1\leq j\approx m}{max}\{c_{ij}\}$$

Equation 1: showing task selection order

If tasks having equal cost ratio then any one of the tasks is selected randomly. Alternative policies can also be applied for tie-breaking such as the task having overall communiqué cost with the other tasks is minimum can be selected. Linear array is considered here Tasg { } = $\{(t_i, p_j) : t_i \in T, p_j \in P\}$ to store the allotted tasks with respect to assigned processors initially, it is supposed that the linear arrays Tasg {} is unfilled.

Step 3: Assignment of the tasks to the processors

In order to form best use of the resources, it becomes essential to maximize the turnout by allocating the tasks to processors in such some way that the allotted load on all the processors ought to be balanced and to attenuate the lay to rest tasks communication by assignment tasks to same processor the maximum amount as attainable. When creating cluster by cluster making algorithmic program (TCM) we tend to assign these task cluster to best processors determined by Hungarian technique. Any we tend to establish that dynamically reconfiguring hardware and software system in response to work changes has the potential to enhance performance. it's attainable to predict that configurations can outmatch at any given time.

Retrieval Number: K19740981119/19©BEIESP
DOI: 10.35940/ijitee.K1974.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

We tend to extend this prediction capability to form precise numerical predictions of the quantitative modification in performance once the system is switched to various configurations .Automatic adaptation hardware and software system reconfiguration will considerably improve overall system performance once workloads vary. The fast development of reconfigurable servers indicates that they're going to become additional normally used. As this hardware is deployed, it are often used for distributed applications wherever analytic the front and back ends is fascinating, however wherever additional hardware is extra. In these cases, the server can want some type of ability to modify ever-changing workloads. TPC-W benchmark, Weka package-implements several machine-learning algorithms for precisely for self-reconfiguration of C.P.U., WIPS are some C.P.U. (processor) self-reconfiguration set ups utilized in distributed computer system in conjunction with on-line cloud conjugation portals like AWS, Sauce lab, Bootstrap etc.

Flow Chart for Proposed method:

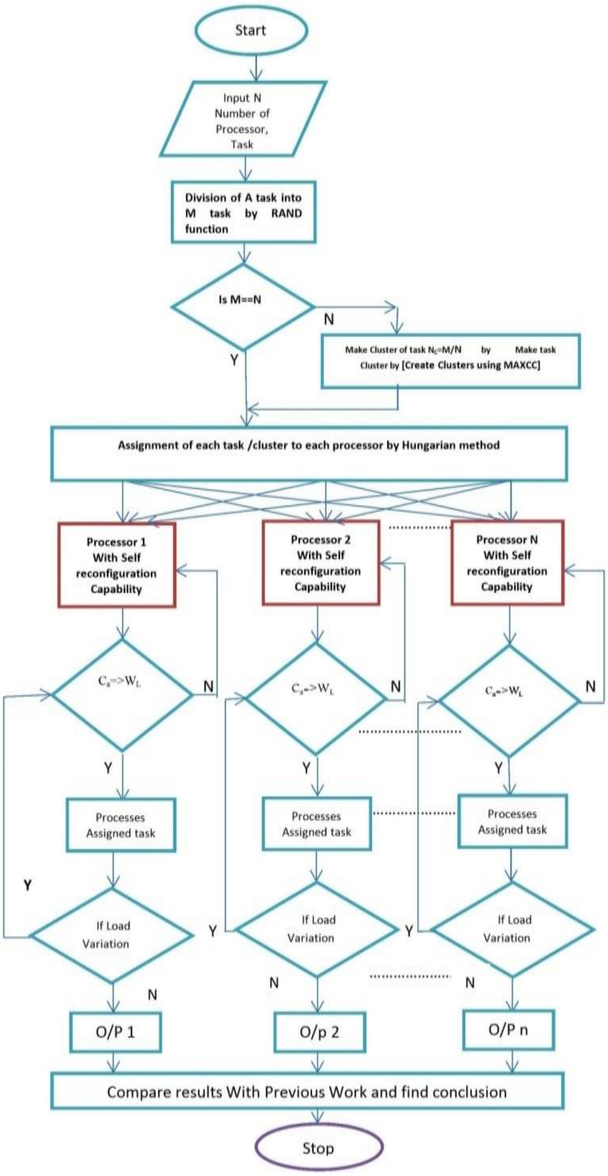

Fig 1: Flow chart of proposed method

VI. ALGORITHM:

I. Input TS , N, M, *PSR(,)// N Dual Mode processors Normal and self-reconfigurable //*

II. Find M random division of Task by RAND function.

III. As M>>N calculate N_c Number of cluster) =M/N.

IV. Store the upper diagonal cost of IMCC and their position MXCCM (,) of order m (m-1)/2x3.

V. Arranging the All Pair Matrix in descending order on the basis of cost between task pairs.
MAXCC [][]=desc Order (APM[][])

VI. Make task Cluster by [Create Clusters using MAXCC] Initialize r=1,s=1;
Repeat r<=n Repeat s<=m Cr =MAXCC[s][1];
// Cr is initial cluster i.e. 1st task of MAXCC is initialize to Cr
p=1;
p counter variable, initialize the condition to open new cluster
Cs=Adjacent of Cr, & yet not assign to any clusters;
Repeat p<=sizeof Cluster;
Cr ← Cr U Cs;
P++; // increase the # task in Cluster
End loop
s++;// Move to next task
End loop
r++; //Open new cluster
End loop
}.

VII. Assigned each task cluster to each processor by Hungarian method.// An optimal method for assigning task to optimal processors//

VIII. If C_a=>W_L { C_a Processor capacity , W_L Work Load}
Then Use Normal Mode and Process Assigned task
Else
Processor Reconfigure itself by processor Self–reconfiguration technique as per need.
a. Hardware reconfiguration
b. Software reconfiguration

IX. Process Assigned task by Respective processor

X. If Load varies by any other factor internal or external.
Then go to step VIII to Step IX.

XI. Store the processed task with their position on processor in task processor matrix FEC.

XII. Now calculate Execution cost, Communication cost, Throughput and service rate by Final execution cost matrix.

XIII. END

VII. RESULT AND DISCUSSION:

To check the fruitfulness of projected method, we considered a distributed computing example which consists of a set of three processors connected by an arbitrary network and task size (TS). We have taken processors per bit service rate (PBSR), and inter tasks communication cost (ITCC) randomly as below:-

Input:

Task size TS, Number of Processor =N

TS () = 2376 it is assumed

We divide this task size by RAND Function into M Module of random size.

Random Number = (RAND ( ) % TS) + 1

TS' ( ) =

t1	240	0.106
t2	300	0.132
t3	190	0.084
t4	301	0.133
t5	225	0.099
t6	255	0.112
t7	232	0.102
t8	245	0.108
t9	280	0.123

Table: 1 showing task division TS ( )

t1	240										
t2	300										
t3	190										
t4	301										
t5	225	x	0.106	0.132	0.084	0.133	0.099	0.112	0.102	0.108	0.123
t6	255										
t7	232										
t8	245										
t9	280										

Table: 2 showing Multiplication of task division and per bit processor rate of that task division

Now M=TS=9 so number of task to be executed are M=9.As M>>N we calculate Nc Number of cluster = M/N.

Number of cluster to be made=number of processor. N_C =M/N=9/3=3 Number of task cluster. Now we make N_C = 3 Cluster from M= 9 task, each cluster must have three task. Now calculate inter module communication by obtain inter module communication matrix.

$$IMCC(i,k)=\begin{array}{c|cccccc|}
 & t_1 & t_2 & t_3 & . & . & t_m \\
t_1 & cc_{11} & cc_{12} & cc_{13} & . & . & cc_{1m} \\
t_2 & cc_{21} & cc_{22} & cc_{23} & . & . & cc_{2m} \\
t_3 & cc_{31} & cc_{32} & cc_{33} & . & . & cc_{3m} \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
t_m & cc_{m1} & cc_{m2} & cc_{m3} & . & . & cc_{mm}
\end{array} \; m\times m$$

IMCC(,)

	t1	t2	t3	t4	t5	t6	t7	t8	t9
t1	0.000	31.746	20.106	31.852	23.810	26.984	24.550	25.926	29.630
t2	31.746	0.000	25.132	39.815	29.762	33.730	30.688	32.407	37.037
t3	20.106	25.132	0.000	25.216	18.849	21.362	19.436	20.525	23.457
t4	31.852	39.815	25.216	0.000	29.861	33.843	30.790	32.515	37.160
t5	23.810	29.762	18.849	29.861	0.000	25.298	23.016	24.306	27.778
t6	26.984	33.730	21.362	33.843	25.298	0.000	26.085	27.546	31.481
t7	24.550	30.688	19.436	30.790	23.016	26.085	0.000	25.062	28.642
t8	25.926	32.407	20.525	32.515	24.306	27.546	25.062	0.000	30.247
t9	29.630	37.037	23.457	37.160	27.778	31.481	28.642	30.247	0.000

Table: 3 showing Inter module communication cost matrix

Now we calculate execution cost matrix ECM (),Here we used dual mode processor (Normal mode and Real-time self –reconfiguration) to maximize performance under a variable workload If Ca=>WL { Ca Processor capacity , WL Work Load}Then Processor will processed Assigned task to it. Otherwise on the other hand Processor Reconfigure itself by CPU Self–reconfiguration technique. According to demand it change its configuration as the workload varied. Automated adjustive hardware and software reconfiguration will considerably improve overall system performance once workloads vary. The server may want some type of ability to handle ever-changing workloads. Many machine-learning algorithms has been implemented by suing TPC benchmark and Weka package exactly for self-reconfiguration of CPU, WIPS along with online cloud servicing portals like AWS, Sauce lab, Bootstrap etc. are some CPU (processor) self-reconfiguration set ups used in distributed computing system. Table 4 shows the PSR of simple heterogeneous servers and self-reconfigurable heterogeneous server.

$$\begin{array}{c|c|}
t1 & 240 \\
t2 & 300 \\
t3 & 190 \\
t4 & 301 \\
t5 & 225 \\
t6 & 255 \\
t7 & 232 \\
t8 & 245 \\
t9 & 280
\end{array} \times \begin{vmatrix} 0.5952375 & 0.765957899 & 0.67889 \end{vmatrix}$$

PSR () =

p1	0.5951	Self-reconfig urable Processors	p1	As per Demand
p2	0.7658		p2	As per Demand
p3	0.6789		p3	As per Demand

Retrieval Number: K19740981119/19©BEIESP
DOI: 10.35940/ijitee.K1974.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

		p1	p2	p3
EC(,)	t1	140.433	181.406	162.933
	t2	175.541	226.757	203.666
	t3	111.176	143.613	128.988
	t4	176.126	227.513	204.345
	t5	131.656	170.068	152.749
	t6	149.210	192.744	173.116
	t7	135.752	175.359	157.502
	t8	143.359	185.185	166.327
	t9	163.839	211.640	190.088

Table: 4 showing processor per bit service rate PSR

Table: 5 showing Execution cost matrix

Horizontally add intermodule communication cost as by finding the addition of each row of IMCC matrix.

t1	214.603
t2	260.317
t3	174.088
t4	261.082
t5	202.679
t6	226.329
t7	208.268
t8	218.584
t9	245.432

Table 6: Showing upper diagonal cost of IMCC

Store the upper diagonal cost of IMCC and their position MXCCM (,) of order m (m-1)/2x3

Arranging the All Pair Matrix in descending order on the basis of cost, we get MAXCCM (,) as

2	4	39.815		6	8	27.546
4	9	37.160		1	6	26.984
2	9	37.037		6	7	26.085
4	6	33.843		1	8	25.926
2	6	33.730		5	6	25.298
4	8	32.515		3	4	25.216
2	8	32.407		2	3	25.132
1	4	31.852		7	8	25.062
1	2	31.746		1	7	24.550
6	9	31.481		5	8	24.306
4	7	30.790		1	5	23.810
2	7	30.688		3	9	23.457
8	9	30.247		5	7	23.016
4	5	29.861		3	6	21.362
2	5	29.762		3	8	20.525
1	9	29.630		1	3	20.106
7	9	28.642		3	7	19.436
5	9	27.778		3	5	18.849

Table: 7 showing All Pair Matrix in descending order on the basis of cost

Now we make cluster of task, in this case we make 3 clusters and each cluster have 3 tasks. Applying the Task Cluster Making (TCM) Algorithm, we get three tasks clusters and modified ETM (,) matrix as:

1. Cluster C1= (t2+t4+t9)
2. Cluster C2= (t6+t8+t1)
3. Cluster C3= (t3+t5+t7)

Cluster size matrix

CS () =		
	C1	766.801
	C2	659.456
	C3	585.030

Table: 8 showing cluster Size matrix

Now assign each cluster to each processor by Hungarian method. After assigning the each cluster to each processor again it checked dynamically if load vary at any processor by any other factor then Processor will reconfigure itself by CPU Self–reconfiguration technique. Now the processor processed the assigned task cluster to it. Store the processed task with their position on processor in task processor TP matrix.By Hungarian method on following multiplication of task cluster matrix and Self reconfigurable processor (when these are in normal mode)bit rate matrix we find optimal task execution cost in term of FEC.

C1	766.801		P1	P2	P3
C2	659.466	x	0.5851	.0.7558	0.6789
C3	585.03				

Normal Mode		P1	P2	p3
FEC(,)	C1	515.506	665.911	598.099
	C2	433.002	559.335	502.376
	C2	378.584	489.040	439.240

Table: 9 showing final execution matrix

With Use of task processor matrix we find Execution Cost EC, Communication Cost CC, Throughput, Service rate for each processor.

Normal Mode	EC	CC	TOTAL	Throughput	Service Rate
p1	515.506	280.130	795.636	0.003771	0.001257
p2	489.040	249.510	738.550	0.004062	0.001354
p3	502.376	221.640	724.016	0.004144	0.001381

Table: 10 Showing Execution cost EC, Communication cost CC, Total Cost, Throughput and Service rate with respect to Processor P1, P2, P3(When processors are in normal mode)

		Self-reconfigurable Processors		
		p1	p2	p3
C1 C2 C2	x	.5151	.7058	.6279

Retrieval Number: K19740981119/19©BEIESP
DOI: 10.35940/ijitee.K1974.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

If Ca=>WL { Ca Processor capacity , WL Work Load}Then Processor will processed Assigned task to it. Otherwise on the other hand Processor Reconfigure itself by CPU Self–reconfiguration technique. According to demand it change its configuration as the workload varied. Automated adjustive hardware and software reconfiguration will considerably improve overall system performance once workloads vary.

Self-reconfigurable Processor

		P1	P2	p3
FEC(,)	C1	485.506	615.911	558.099
	C2	411.002	511.335	498.376
	C2	314.584	467.040	412.240

Table: 11 showing final execution matrix

With Use of task processor matrix we find Execution Cost EC, Communication Cost CC, Throughput, Service rate for each self-reconfigurable processor.

Self-reconfigurable Processors	EC	CC	TOTAL	Throughput	Service Rate
p1	485.506	251.120	736.626	0.0039821	0.001357541
p2	467.040	239.520	706.56	0.0041750	0.00141530
p3	498.376	218.598	709.974	0.0045746	0.001408502

Table: 12 Showing Execution cost EC, Communication cost CC, Total Cost, Throughput and Service rate with respect to self-reconfiguration processor p1, p2, p3.

Finally by comparison table 14 shows the throughput and service rate for processor p1, p2, p3 on the other hand table 15 shows throughput and service rate for self-reconfigurable processor p1, p2, p3.

Normal Mode of Processor	Throughput	Service Rate
p1	0.003771	0.00125686
p2	0.004062	0.00135400
p3	0.004144	0.00138118

Table 13 shows the throughput and service rate for processor p1, p2, p3.

Self-reconfigurable Processor	Throughput	Service Rate
p1	0.0039821	0.001357541
p2	0.0041750	0.00141530
p3	0.0045746	0.001408502

Table: 14 showing the comparison of throughput and service rate with respect Self-reconfigurable Processors to processor p1, p2, p3

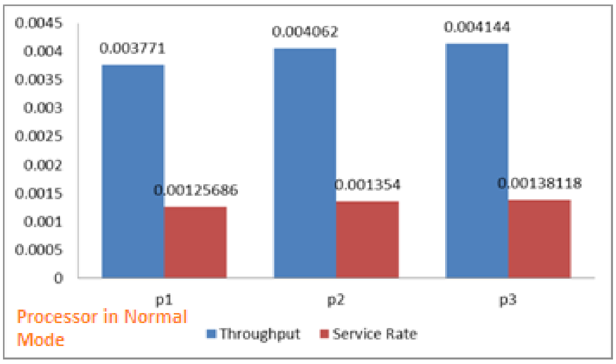

Fig: 2 Graph showing service rate and throughput with respect to when processor p1, p2, p3 in normal mode.

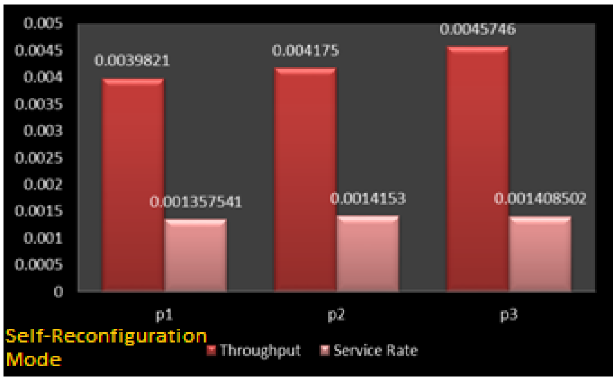

Fig: 3 Graph showing service rate and throughput with respect to self-reconfiguration processors p1, p2, p3

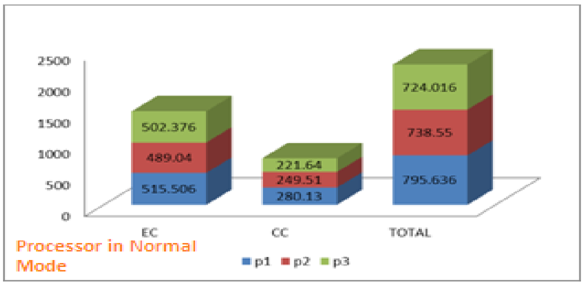

Fig: 4 Bar Chart of processor p1, p2, p3 in respect to execution cost EC, Communication Cost CC, Total Cost (EC+ CC).

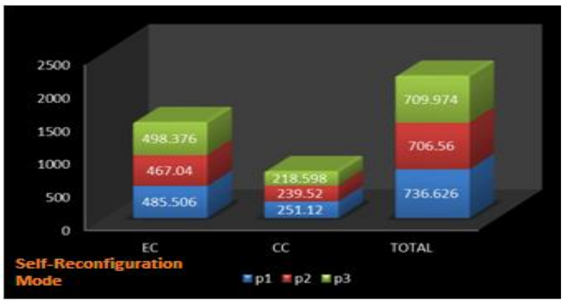

Fig: 5 Bar Chart of self-reconfiguration processor p1, p2, p3 in respect to execution cost EC, Communication Cost CC, Total Cost .

Retrieval Number: K19740981119/19©BEIESP
DOI: 10.35940/ijitee.K1974.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

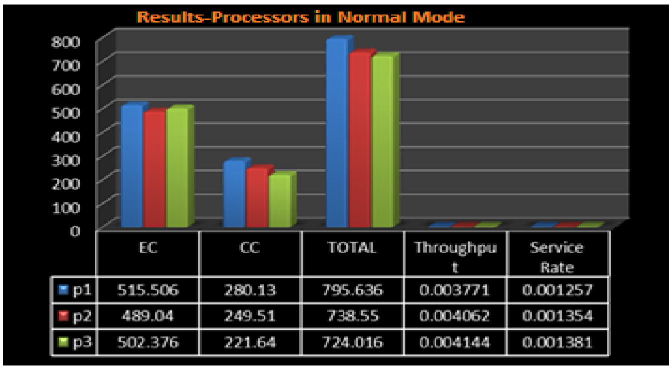

Figure: 6 (a) Showing comparison of EC, CC, and Total Cost, throughput and service rate for processors p1, p2, p3.

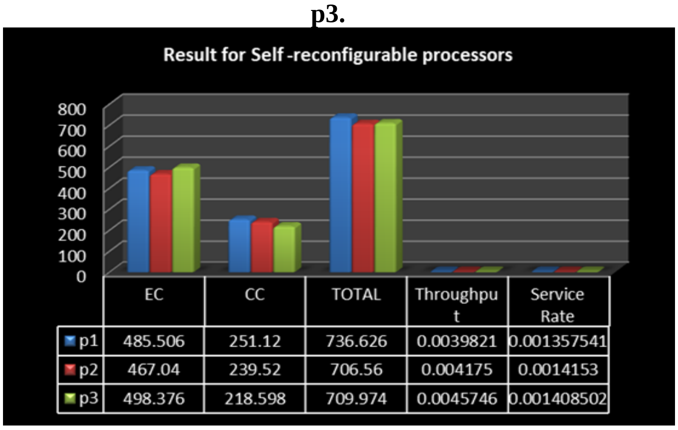

Figure: 6 (b) Showing comparison of EC, CC, and Total Cost, throughput and service rate for self – reconfigurable processors p1, p2, p3.

VIII. CONCLUSIONS

In this paper we deal with a hybrid method of best possible allotment of task and use of dual mode processor in order to optimal utilization processors capacity and augment the performance of distributed systems. FEC matrix shows that three tasks are executing on processor p_1, three tasks are executing on p_2 and three tasks are executing on p_3. With varied load the performance of each processor is differ while we have taken all three processor heterogeneous having different PSR and all processor have same reconfiguration techniques as the load vary by any factor they have capability to reconfigure itself according to varying load from tied up such service providers . Here we used fully adaptive task scheduling applied on dual mode processor (Normal and self-reconfiguration mode) to find the optimal use of processor resources in dynamic condition. So through this paper we projected an algorithm which is the mix hybrid method of effective task allocation to processor involved in computing and self-reconfiguration of those processors as per need of computing. We **obtained optimal** Execution cost EC; Communication cost CC, Total Cost, Throughput and Service better than previous approaches as an outcome of organized processors in heterogeneous distributed computing system.

REFERENCES:

1. SHATZ, S/M., and WANG, J.P., “Introduction to distributed software engineering”, Computer, 1987, 20 (10), pp. 23-31.
2. Prashant Singh Yadav , Sunil kr Singh , Pankaj Sharma.,“ Self-Reconfiguration of CPU- Enhancement in the Performance”, in IJECSE , V1N2 2010 , pp. 166-172
3. D.F. Baca, “Allocation Tasks to Processor in a Distributed System,” IEEE Trans. On Software Engineering, Vol.15 pp.1427-1436, 1989.
4. S.H. Bokhari, “Dual Processor Scheduling with Dynamic Re-Assignment”, IEEE Trans. On Software Engineering, Vol.SE-5 pp. 341-349, 1979.
5. T.L. Casavent, and J.G. Kuhl, “A Taxonomy of Scheduling in General Purpose Distributed Computing System”, IEEE Trans. On Software Engineering, Vol.14 pp.141-154, 1988.
6. W.W. Chu, “Optimal File Allocation in a Multiple Computing System”, IEEE Trans. On Computer, Vol.C-18 pp.885-889, 1969.
7. O.I. Dessoukiu-EI and W.H. Huna, “Distributed Enumeration on Network Computers,” IEEE Trans. On Computer, Vol.C-29 pp.818-825, 1980.
8. Ma, P.-Y. R., Lee, E. Y. S., & Tsuchiya, M.. A Task Allocation Model for Distributed Computing Systems. Computers IEEE Transactions on, C-31(1), 41-47, 1982.
9. GamalAttiya and YskandarHamam**,** **“**Task allocation for maximizing reliability of distributed systems: A simulated annealing approach”, Journal of Parallel andDistributed Computing, Volume 66, Issue 10, Pages 1259-1266, 2006
10. Yadav P. K.**,** Bhatia K. and GulatiSagar ,Reliability Evaluation of Distributed System Based on Failure Data Analysis, International Journal of Computer Engineering,Vol 2(1) pp.113-118, 2010.
11. Singh M.P.,KumarHarendra ,Yadav P.K.” An Efficient Algorithm for Optimal Tasks Allocation through Optimizing Reliability Index in Heterogeneous Distributed Processing System” in “3rd International Conference on Quality, Reliability and Infocom Technology” held at Indian National Science Academy, New Delhi –India, December 2-5.2006.
12. J.B. Sinclair, “Efficient computation of optimal assignments for distributed tasks,” J. parallel Dist. Comput., vol.4, pp.342-362, 1987.
13. Peng, Dar-Tezen; K.G. Shin, Abdel, T.F. Zoher, “Assignment Scheduling Communicating Periodic Tasks in Distributed Real-Time System,” IEEE Trans. On software Engg, vol.13, no.12, pp.745-757, 1997.
14. G. Sagar, and A.K. Sarje, “Task allocation model for distributed system,” Int. J. system science, vol. & no. 22, 9, pp. 1671-1678, 1991.
15. Vinod Kumar, M.P. Singh, P.K. Yadav, “A fast algorithm for allocation task in distributed processing system,” proceedings of the 30th Annual convention of computer society of India, Hyderabad, pp.347-358, 1995.
16. Vinod Kumar, M.P. Singh, P.K. Yadav, “An efficient Algorithm for allocating tasks to processors in a distributed system,” Proceedings of the Nineteenth National system conference (NSC-95), PSC College of technology, Coimbatore, organized by system society of India, New Delhi, pp. 82-87, 1995.
17. Vinod Kumar, P.K. Yadav, and K. Bhatia, “Optimal Task Allocation in Distributed Computing Systems Owing to Inter Task Communication Effects.” Proc. Of CSI-98 : IT for the New Generation, pp.369-378, 1998.
18. Vinod Kumar, M.P. Singh, P.K. Yadav, “An efficient Algorithm for Multi-Processor scheduling with Dynamic Re-Assignment,” Proceedings of the Sixth National Seminar on Theoretical Computer Science, BanasthaliVidya Pith, pp. 105-118, 1996.’
19. Yadav, P. K., Kumar, Singh, M. P, and Harendra Kumar “Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with dynamic Reassignment” International Journal of Computer System, Network and Communication, Vol. (2008), pp. 1-9 2008.
20. ElsadeA.A.,Wells B.E. “A Heuristic Model for Task Allocation in Heterogeneous Distributed Computing System” The International Journal of Computers and Their Applications. Vol.6, no.1, March 1999.
21. Yadav, P. K., Singh Jumindera and Singh, M. P “An efficient method for task scheduling in computer communication network” , International Journal of Intelligent Information Processing Vol. 3(1) pp 81-89, 2009
22. Yadav P.K., Singh M.P., Kumar Avanish and AgarwalBabita, “An Efficient Tasks Scheduling Model in Distributed Processing Systems Using ANN”, International Journal of Computer Engineering, Vol 1(2) pp.57-62, 2009.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

23. Singh.M.P, Kumar Avanish, Yadav, P.K. and Krishna Garima, " Study of Load Distribution in fully connected client server Network using feedback neural network architecture", Journal of Engineering and Technology Research, Vol 2(4) pp. 58-72, 2010
24. Kumar Avanish, Yadav P.K., and Sharma Abhilasha "Analysis of Load Distribution in Distributed Processing Systems Through Systematic Allocation Task" IJMCSIT "International, Journal of Mathematics, Computer Science and Information Technology", Vol 3 (1), ,pp.101-114, 2010
25. Yadav, P. K., Kumar, Avanish and Singh, M. P., "An Algorithm for Solving the Unbalanced Assignment Problems", International Journal of Mathematical Sciences, Vol. 12(2), pp. 447-461 2004.
26. M. I. Daoud and N. Kharma, "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems," Journal of Parallel and Distributed Computing, vol. 68, no. 4, pp. 399 – 409, 2008.
27. S. Jin, G. A. Schiavone, and D. Turgut, "A performance study of multiprocessor task scheduling algorithms," The Journal of Supercomputing, vol. 43, no. 1, pp. 77–97, 2008.
28. H. D. Karatza, "Scheduling strategies for multitasking in a distributed system," in Annual Simulation Symposium, 2000, pp. 83–90.
29. Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," ACM Comput. Surv., vol. 31, pp. 406–471, December 1999.
30. T. Davidovi´c and T. G. Crainic, "Benchmark-problem instances for static scheduling of task graphs with communication delays on homogeneous multiprocessor systems," Computer. Operation. Res., vol. 33, pp. 2155–2177, August 2006.

AUTHOR PROFILE

Prasant Singh Yadav `completed bachelor of Engineering in Computer Science & Engineering from Dr. B.R Ambedkar University Agra in 2005, Master of Engineering from NITTTR (an Institute of national importance) Chandigarh Punjab University Chandigarh (U.T) India in the year 2011, and currently Ph.D. Scholar in Computer in Computer science & Engineering at Dr. A P J AKTU Lucknow, UP, India Having more than 12 year academic experience. Author of more than 65 research papers published in Different national and international journals, conferences and proceedings. He has been member of more than 12 international / national Educational Society.

Dr. Pradeep kumar Yadav, Currently working as Principal Technical Officer CSIR-CBRI, Roorkee & Associate Professor, Mathematical and Information Sciences at (AcSIR), completed his M.Sc mathematics in 1983 from Grukul kangri university hardwar U.P India, received his Ph.D. in 1996 in Distributed Computing System Grukul kangri University hardwar U.K India Uttar Pradesh, India; He has more than 20+ years of work experience in teaching 10+ experience in R&D. He has been published more than 69 technical research papers in numerous international journal and conferences; He also more than 42 research papers published in various National Journals, conferences proceedings. He has been Life Member of International Academy of Physical Sciences; Indian Society for Wind Engineering, Computer Society of India (CSI) and has is awarded many research awards and working with many research project grants in India.

Dr. K. P Yadav, Currently working as Vice chancellor SANGAM University Bhilwara, Rajasthan, India., completed his B.E CSE in 1996 from KNIT Sultanpur and M.Tech BITs Mesra, Ranchi, India , obtained his Ph.D. in 2009 from Corllins University, CA (USA), and Ph.D from BIET Jhansi/Bhagwant University Ajmer in 2012 India; He obtained Doctor of Science (D. Sc.) from MNIT Jaipur/ OPJS university Churu Rajsthan India. He is having vast experience of 24+ years. He has been published more than 85 technical research papers in various international journal and conferences; He also more than 33 exploration papers published in various National Journals. He has written more than 12 books in engineering in reputed publication of India.

Dr.Sunil K Bharti, Currently working as Assiciate Professor in Galgotia College of Engineering, Greater Noida Uttar Pradesh, India. He Completed his MCA, M.Tech and Ph.d CSE from Jawahar Lal Nehru University Delhi. He is having vast experience of 7+ years. He has been published more than 20 technical research papers in various international journal and conferences; He also more than 5 exploration papers published in various National Journals. He has written many books in engineering in reputed publication of India.

Retrieval Number: K19740981119/19©BEIESP
DOI: 10.35940/ijitee.K1974.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)