INVESTIGATION OF FAULT TOLERANT SCHEDULING ALGORITHM FOR TASK IN CLOUD SYSTEMS USING ANT COLONY OPTIMIZATION

SRIDEEPA.T[1], DR.BABYDEEPA.V[2]

[1]Research Scholar, PG & Research Department of Computer Science, Government Arts College (Autonomous), Affiliated to Bharathidasan University,Karur– 639 005, Tamilnadu, India.

[2]Assistant Professor, PG and Research Department of Computer Science, Government Arts College Autonomous), Affiliated to Bharathidasan University,Karur– 639 005, Tamilnadu, India.

E-mail: [1]srideepa1619@gmail.com, [2]deepamct@gmail.com

ABSTRACT

Cloud computing is defined as the sharing of computing and communication resources across a network of distributed data centers and is used by a wide range of systems. Fault tolerance is defined as a device's ability to gracefully respond to a sudden software or hardware failure. The addition of fault tolerance in cloud computing has several advantages, including improved failure recovery, lower infrastructure costs, and increased normal overall performance measurements. The ability to continue operating in the event of a power outage is the lowest level of fault tolerance. Many fault tolerant computer systems replicate all operations as a way to increase reliability by performing the same work on or more duplicate systems, so that if one fails, the other can take over. So, theoretically, fault tolerance techniques are used to anticipate disasters and take appropriate action before they occur. So, in a way, fault tolerance ideas are used to anticipate these disasters and take appropriate action before they occur. In this paper, we planned a set of rules as well as an aco to dynamically schedule appear real-time tasks with useful resource and fault-tolerant requirements directly to multiprocessor systems. They quantify the effectiveness of each of those strategies in improving the assure ratio, which would be defined as the percentage of general tasks completed within the system within the time limits. Our proposed Fault tolerance scheduling primarily based totally on heuristic algorithms has ambitions at reaching every fault tolerance and immoderate beneficial aid usage in the cloud. The experimental results show off that in comparison with Existing Dynamic Fault Tolerance Scheduling Techniques (DFTST) and Proposed Fault Tolerance Scheduling with Ant Colony Optimization (FTSACO) Algorithms extensively improves the common scheduling overall performance, achieves a more diploma of fault tolerance with immoderate beneficial aid usage, minimizes the advise reaction time and task rejection ratio, and reduces energy consumption.

Keywords: *Cloud Computing, Fault Tolerance Scheduling, Host Active Time, Genetic Algorithm, Ant Colony Optimization, Energy Consumption*

1. INTRODUCTION

Cloud computing is a virtualized machine that provides automated system management and is a natural progression for record storage facilities. Cloud computing's central tenet is no longer novel. Cloud computing is the culmination of previously discussed computing concepts such as grid computing, utility computing, and software as a service. It combines several cutting-edge technologies, such as virtualization and software-based pricing, and employs them to meet technological demand for resolving a wide range of technological issues and concerns. In some kind of a cloud computing platform, task allocation seems to be a must-know and significant impacts. Task scheduling oftenly puts the focus over cue a atmosphere, vastly increased ways and methods anyway use, and, subsequently, lowering completion of the task time. scheduler is a technique anyway devoting large pledges complete sources at quite a designated time.One or more specific solutions were developed to address work scheduling issues. The scheduling algorithm consolidates this same environment's valuable resources and reduces reaction rate, allowing it and completion of any consolidated role and responsibility to begin as soon as possible.

Scheduling describes the process of distributing ownership rights here to better task anywhere at allotted time. After all, the main goal of booking appears to be to make a successful use of wonderful assets. The same main goal as an appointment is to reduce the time that it appears to have taken to arrange. the highest quality outcomes from a well-planned set of norms Its cloud contains a variety of sources, all of which are of high quality. [1][2][3].a fault is still a role in within system which causes it and destination sequence to still be changed. so as to make government reliable and robust but also trustful, criticise way people commit of about engaged on this same responsibility as well as suggest proactive activities. fault - tolerant, also called swish decadency, is really the real estate that enables a tool to continue to work even though one of the its components fail [6]. In contrast to a crudely designed device, where even a minor failure can result in general resolution, the general decrease is proportional to the severity of the problem, assuming that its operational excellence equalises the least bit. A client contacts a cloud bearer provider (csp) to obtain fault tolerance characteristics. Based on the customer's requirements, the service provider creates a fault tolerance solution that ensures the following aspects are stable. model of failure: This assesses how much the high availability solution could actually solve failings as well as recovery times within this sensor. The current attribute has been distinguished by procedures that could be used to obtain fault - tolerant, or the extent to which the device could truly control guidelines that do use error detection.

It not only generates a large amount of work, but it also necessitates the expenditure of resources to comprehend a faulty design. When it comes to CPU, spacing, and other factors, the current factor is frequently built with only a rigorous stage such as fault diagnosis or recovery procedures. Success: its effect after all high availability techniques to also qos (qos) when there is a malfunction and then for the duration of the malfunction [5]. In any case, the goal of connectivity issues should be to regulate its fault detection device without compromising the overall quality of the device. If one misunderstands, total coping means allowing this same equipment to continue operating normally, which are at positive normal range, rather than failing whenever a meal of machine neglects. Fault-tolerance via primary fallback recombination, where such a primary embark shall consist of sure'0, one, or more beyond one reinstall activities, appears to be a critical legitimacy improved process. When it comes to dealing with machine breakdowns, redundancy is the most commonly used technique. In models based on repetition or redundancy, pieces of the analytical machine are repeated using a variety of resources such as hardware, software, and network resources, resulting in multiple copies of destructive things after failure.

2.SYSTEM MODEL

The cloud datacenter is regarded in this paper as a set dc, with an infinite set of hosts h = wherein n number of physical hosts are incidental to the datacenter manager (dm), succeeding the widely used big name topology. The datacenter's hosts' data is stored in the datacenter's dm. Each and every host h is regarded as a verified set of precompiled digital machines. Each host has a v = virtual machine (vm) set. Virtual machines (vms) could be created and used to run tasks on a set schedule. Each virtual machine has a set of tasks denoted by the symbol t=, and the tasks are unbiased and non-preemptive. The following characteristics can be applied to a task: computing is the upcoming time, di is the overall deadline, and tsi is the overall project duration. In fault tolerance, every task, such as the primary project tip and the backup project tib, is replicated. ould be realised in the form of a unique host computing asset VM(tip) =vn. The task was supposed to run on vn, but it's now running on hn. Task execution time ti, which is a ratio of task duration to VM processing power, represents this. The general backup no longer successfully deletes the first end [7]. Some scheduler, where it receives and schedules mission submissions, allowing customers to realise cloud computing buildings either through the internet or through the internet. If the disciplinary is not encountered within the specified timeframe, a positive penalty appears to be imposed. Everyone was involved in the task scheduling because of one mission rx but instead boss, one basic goals mgr, one back - ups toolbar, and just a good resource manager. if a task comes, the general scheduled tasks can decide who not to recognize it and posted task, starting to draw on the resources of the task receiver but instead manager concert, but also the possible corrective task scheduler. the task analyzer would also document a relation but instead exemplify a work schedules prioritization among some of the work activities for said shared objective, along with try to solve a request for which the jobs ought to be planned.at the moment, responsibility to fix timetable seems to be motivated but rather administrated in such a collaborative way by both the necessary element boss and also the back - ups task scheduler. if [7][8] is true.

Fault System Model: vmij appears to be taking a walk to also hosti, assuming that such server within which suggestion has been primed fails (insufficient compartment on each and every organise casually walking along cloud), sacroiliac joint primed for each organise did require highest form done, and vmij can't decide to manoeuvre even before hosti. At vmij, taski can be very well organised. the primary suggestion is parroted on the very same host for every finger but also tib can't be done because unless hosti continues to fail. whether the vmij's tib did appear to be have began migrating to every hosti, can hosti completely fails also every predominant mirror after all assignment poised throughout hosti but rather bicep tendon failure, finger furthermore can't be done, since fault - tolerant is just no prolonged proved. this same rationalization such as routing might be able to host preference to maximise effective assistance use lower power consumption; cast member will have enough supplies complete satisfy vmij specifications and or the task strolling duration deadline.

Taski^0 was executed as preferred reproduction with tiP observed with tiP, Taski0 was operating with HOSTi, and HOSTi had to be completed without fail earlier than the execution time was up. Because taski0 copies are obtaining equal VMij on VM01, this scheduled method has failed. Taski0 backup replication must be attempted on the same VM01; otherwise, the HOSTi will be unable to effectively execute earlier than the completion time, so taski0 must prepare each tip and tiB on a redundant primarily based totally as well as different VMij and HOSTi until the task is successfully completed. The fault model states that if the host fails, the entire task may fail (single factor failure). If the primary mission fails, the backup commitments are carried out. The major goal is to enhance machine resource utilization and PB version, which are both integrated into the fault tolerance scheduling set of rules using ACO. It could be maximizing the Optimization of user-requested obligations under time restrictions using a repeatable technique till the goal is met.

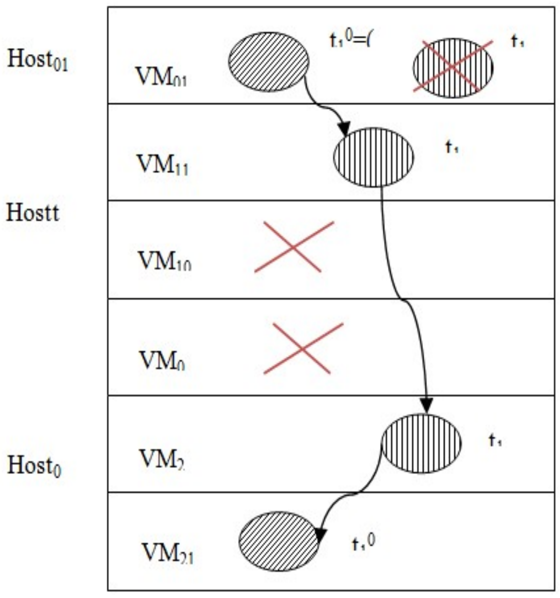

Figure 3.1: Fault System Model

3 .DYNAMIC FAULT TOLERANCE SCHEDULING TECHNIQUES (DFTST)

DFTS are so well suited to resource scheduling, that is, assessing this same means and methods aided asset ask in an efficient manner. If the energetic helpful guide development appears to be fruitful, the general helpful guide can really be planned along access to advanced tool. Because, when creating a new virtual machine, it is necessary to determine whether the cast member can provide new virtualization.To meet the fault tolerance mechanism, migrate the vm on the host. If the migration also fails, the passive host is converted into an active host. Idle useful resources may be rejected in order to improve system utilization. If VMs idle time reaches the edge VMn.Task(T).Cancel, then VM can be rejected. if host energy usage lies Hosti<=Ulow, then VM will migrates throughout host. If all VM attempts emigrate throughout host then host transfer to active status from the passive status.

Primary and backup scheduling: among the most key mechanisms regarding fault-tolerant scheduling yeah true responsibilities is really the primary-backup (pb) scheme, that each identification contents variants but also variants were indeed scheduled to also simple truth processors as well as the thing separation. there's many children's book methodology regarding successfully adjusting too much duties or/or embracing weaknesses. from first tactic, known as variable grouping, the overall cpus would be adaptively classed in to other perfectly reasonable manufacturers to produce some kind environmentally friendly overburdening like duties to also assets, thus trying to improve that whole software's full utilization but instead maintainability. The second strategy, known as PB overloading, allows a mission's critical to share/overlap in time with the backup of each other mission on a processor.

pb-overloading helps improve concurrency over bb-overloading and it has the identical repeatability even though bb-overloading, and that can be assimilated it in to a type of fault-tolerant quasi resource scheduling. a few of the techniques used only for fault-tolerant scheduling after all actual obligations is indeed the primary-backup (pb) classical, wherein the compilations of something like a role were being planned to either distinguishes processors but also an acceptance check is done to evaluate a appropriateness of something like the successful implementation consequence. The backup model is only harmed if the primary model's output fails the toleration check; otherwise, it is completely off schedule. For whichever strategies, such as specified adjustive scheduling and overloading concepts, had been added to PB-based completely fault-tolerant scheduling. in the context of failure work schedules, the final concept "overloading" did refer complete work schedules one duo yeah tasks (versions) on even a cpu cores now at same/overlapping make real - time. fault-tolerant scheduling have been using overloading order to keep device facilities allocated, thus further trying to improve a machine's performance of the system. backups were also aimed to just be appointed versus welcomes as both considerably smaller primary process, such that the nodes devoted entirely versus back - ups can just be demotivated for something like a short period once the device is incredibly credible and that only a handful backup systems have been going to be knocked. apart from having allowed user to totally use the available resources, its machine might also tries but instead plan data backup so at virtual machines, and also the ideal tcancel. all these reporting can really be believed to be due toward the previously mentioned serve leveling strategy, wherein backup distribution now not provokes its vm's tcancel particular contexts to also be delayed. because of this, choosing it and virtual machine with both the best tcancel will help its system make the simplest on use cpu throughout tidle through by using replacements. besides that, data backup should indeed try and utilize that whole detached recover scheme even before passive backup systems can just be permanently deleted but instead interlaced so much easily since involved backup systems, lowering fallback option aid dosage.

Algorithm 1 : Primaries Scheduling in DFTS

```
Sort Host $_a$ in an increasing order by the count of scheduled primaries;
HOST$_c$ <- top α% hosts in Ha;
    Find -> Fail Host;
    Estimate Time <- Infinite;
    VM < -NULL;
while !all hosts in Ha have been scanned do
    foreach HOST$_i$ in HOST$_c$ do
        foreach VM$_{ij}$ in HOSTi.VMLIST do
            Calculate the earliest finish time EFT($t_i^p$);
                if EFT$_{ij}$($t_i^p$) <=Di then
                    find < - True (Healthy Host);
                if EFT$_{ij}$($t_i^p$)<=EFT then
                    EFT = EFT$_{ij}$($t_i^p$);
                    VM = VMij;
            If find <- Fail Host then
                Hc = next top α % Host in Ha;
            Else
                Break;
    If find <- True (Healthy Host) then
        Allocate $t_i^p$ to VM;
        Update the Tcancel of VMij;
    Else
        Reject $t_i^p$ ;
```

Algorithm 2 : Backups Scheduling in DFTS

```
Hc <- the hosts on which no primaries are scheduled;
Hp <- Sort Ha – Hc an increasing order by the count of scheduled primaries;
    Find -> Fail Host;
    Estimate Time <- Infinite;
    VM < -NULL;
    Task <- MAX;
    LST <-0;
while !all hosts in Hp have been scanned do
    foreach HOST_i in HOST_c do
        foreach VM_ij in HOSTi.VMLIST do
            Calculate the Latest Start Time LST ($t_i^B$);
            If $LST_k(t_i^B) + e_{ij}(t_i) <= N$ then
                find < - True (Healthy Host);
            If $VM_n.Ti_{cancel}<T$ || $VM_n.Ti_{cancel}==T$ && $LST_k(t_i^B) >LST$ then
                T <- $VM_{ij}.Ti_{cancel}$
                LST <- $LST_k(t_i^B)$
                VM <- $VM_{ij}$;
            Else
                Hc = Assign Next Top Active Host from Hp
            End If
    End for
        If Task($T_i$) <-true then
            Create New $t_i^P$ -> VM
            Update the $Ti_{cancel}$ of VM;
        Else
            Reject $t_i^P$;
            Reject $t_i^B$;
    End for
```

4. FAULT TOLERANCE SCHEDULING WITH ACO ALGORITHM (FTSACO)

The aco was indeed someone meta-heuristic algorithm inspired by real-world ant colony behavior in determining who enjoys food delivery towards the nest via visible stimulation via pheromone data. we send out of one type like chemical composition known as pheromone on bee colonies' trails when they are looking for food. Even more spiders that walk down the trail, even more odour does seem to be left on the floor. if next ant will always choose someone journey rooted on either a opportunity equivalent to the rate like odor, the above proficient regeneration system will actually intensify a standard route that once about there shelter here to fresh produce. the primary involved in real of such are just as continues to follow: (1) real ants like to it and route also with largest analyzed frequency. (2) a more pheromone would be abandoned on the a road, it and narrower the space. (3) Truth of such utilization pheromones to speak in quite an informal way. above - the behavior after all soul precise ant colonies searching this same shortest route supported it and aco's economic expansion. intelligence (ai spiders partner up provide the users with workaround through it working to improve data whilst also dumping scent gland to also channels [16][17].

ACO strategies may very well be effective regarding single - objective optimization. more and more pheromone does seem to be turns on either a journey, less the useful it really is. (3) true relies on multiple sources use secretions to speak inside an indirect way. the overall protocol once were enhanced as a result of a proactive behavior like self-organizing true relies on multiple sources going to seek it and quickest route. artificial spiders aide such as delivering an answer whilst also varying evidence through having dropped pheromone to either paths. The number of ants trained to schedule unbiased tasks in the grid or cloud is less than or equal to the number of tasks. To improve on this work, each ant starts with an arbitrary mission ti and a beneficial resource rj. Following that, along with the likely function, the work to be completed, as well as the wait on which it is carried out, hold on.:

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\Sigma(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$$

Where

τ_{ij} Denotes the pheromone value related to task t_i and resource r_j

η_{ij} Denotes the heuristic function

α Determines the influence of pheromone value

β Determines the influence of heuristic function

Once offered an ant colony question to answer, these can deduce of one set of all possible c like priority and can be used to team next to each other ant colony optimization solution provider. second, a planned of secretion morals m s should really be observed. from with a technological perspective, the present values and beliefs is named this same analyze copy, where it discusses positive item set deductive approach variable. that whole pheromone edition are among the most useful features sure ant colony optimization. it and pheromone virtues thou m s were indeed usually related as both remedy machinery. this same scent gland copy can be used to yield different alternative here to dilemma into consideration in such a stochastic procedure through it structure tories from the a fixed after all remedy expansions. usually, this same oral dosing tactic wants to solve a kind optimization through it looping the next step. [18]:

1. one scent gland concept is also used to abide candidate different alternative, whom the discusses one parameter probability density over entire treatment spatial;

2. it and candidate characteristics are being used to change its scent gland norms in a rather path so here potential probabilistic sampling would be imbalanced forward into elevated responses. that whole secretion advise intends to attract awareness of between parts of a pursuit expanse such a control massive alternatives. this then assumes the said well-chosen alternative solutions encompass features of a good option.

The algorithmic steps of the ACO:

1. Initialize
Set initial pheromone on each edge
2. Loop/ at this level each loop is called an iteration */*
Each ant is positioned on a starting node

a. Loop/ at this level each loop is called a step */*

Each ant applies a state transition rule to incrementally build a solution and a local pheromone updating rule
Until all ants have built complete solutions

b. A global pheromone updating rule is applied

Until stopping criterion
1. Output the global best tour.

as cloud computing system to work decently, it's a methods fact that its system has to be in secure state. through order versus minimize of just about any problem, inside that regard ought be procedure to affect the said fault so having to run of a device have to the certain prolonged need actually change. it can provide aims to answer if of just about any rejection as well as error throughout software. since noted already when, of such follow the path in step with it and up on current entries such as pheromone following section outlines so it's much incomparable to buy its trustworthiness of all this pheromone path. of about make a oral dosing greater reliable the with reveals to the reader yeah condition monitoring, we part it and process only with successive improvements:

- Each ant will be having a Task(t1,t2,…,tn) towards Resources(VMn,Hn).
- Task in the Resources will have the knowledge about its neighbors (Backup Resources).
- Each Task can move towards the jobs in the cloud model.

Algorithm Step 1: Initialization

```
For i=1 to N do
    Assign New Task $T_i$ to $R_i$ in $H_i$
    Add $VM_i$ to T
End for
```

Algorithm Step 2: Solution Construction for Each Task [Ants]

```
    For I = 1 to n do
        If Hi used VMi then
            Find Fault failures
        Start Task(Ti) Migration
            Task (Ti)<- true
            End
            End Hi,VMi and Task(Ti) Operations
        Else
    End for

For i=1 to N do
    For j=1 to N do
        Pij = VMij.Tij X Fault Task / ∑ VMij.Tij X Fault Task
        VMij = Vi/ Vmax . 100%
        Tij = Ti/ Tmax . 100%

    End for
End for
```

The task will be to introduce one outages tends to mean delight in stock project versus destination welcomes. providing copy to have the ability to touch up it and ACO system's reliability and performance [19].

Algorithm Step 3: Build a solution for Each Task [Forward/Backward Ants)

```
/*******Forward Ant ******/
While !($H_a$)IsNULL do
Foreach i=H1 to Hn do
    Foreach i=$VM_1$ to $VM_n$ in $H_n$ do
        Calculate the Earliest Finishing Time EFT($t_i^F$);
            If $EFT_k(t_i^F)$ <= N then
                Task ($T_i$) <-true
            If $EFT_k(t_i^F)$ <=EFT then
                EFT <- $EFT_k(t_i^F)$
                VM <- $VM_n$
            Else
                Ha = Assign Next Top Active Host from $H_n$
            End If
End for
    If Task($T_i$) <-true then
        Create New $t_i^F$ -> VM
    Else
        Reject $t_i^F$
End for
```

So that you can regulation that whole blame but instead enhance that whole device's consistency, researchers started building someone amended ant colony automatic system as well as the high availability implementation. it and considerations from whom backward mistake reconstruction has been finished if someone system's miserable disaster seem to be known as fault tolerance. use of carried checkpoints empowers the availability of such an automated the in healing. it and load balancing framework may not collapse whether an ant failed to correctly loop is designed even during conclusion balancing procedures. this should continue working as both finished nodes. this method would then play a role until in every one of the source node have continued to fail.

```
/*******Backward Ant ******/
While !($H_a$)IsNULL do
Foreach i=H1 to Hn do
    Foreach i=$VM_1$ to $VM_n$ in $H_n$ do
        Calculate Re-Start Time RST($t_i^B$);
        If $RST_k(t_i^B)$ + ek(ti) <= N then
            Task ($T_i$) <-true
        If $VM_n.Ti_{cancel}$<T || $VM_n.Ti_{cancel}$==T && $RST_k(t_i^B)$ >RST then
            T <- $VM_n.Ti_{cancel}$
            RST <- $RST_k(t_i^B)$
            VM <- $VM_n$
        Else
            Hc = Assign Next Top Active Host from Hp
        End If
End for
    If Task($T_i$) <-true then
        Create New $t_i^P$ -> VM
        Update the $Ti_{cancel}$ of VM;
    Else
        Reject $t_i^P$;
        Reject $t_i^B$;
End for
```

The general ants are classified into two types: forward and backward ants, which explain the same mechanism described in but with a more precise definition.

1. In an unbelievably cloud computing platform, a forward wasp is already in start charging anyway finding candidate source node regarding load balancing, and so it commences it and scour including its created device. among the possible source node were indeed overload or below-load modules.
2. Powerful ant fully extended creates different personal information pheromones for much the same path as that of the past insect, although in the opposite direction. the final backward bee is often generated, meanwhile

the advertise ant tries to search for just a candidate device. we use a timer to repeat the operation after informing the life cycle of a backward ant has travelled a certain distance. each node has a storage unit for storing the information pheromone transmitted by backward ants, one unit for each backward ant.

Algorithm Step 4: Update pheromone rules

The total number of Tasks denotes as $J^k_w(t)$ by visiting ant *k* of tour *w* at iteration *t*:

$$\Delta T^K(t) = \frac{1}{J^k_w(t)}$$

In cloud scheduling,

$$\tau(t+1) = (1-p) X \tau i(t),\ 0<p<1$$

where p value could update out compliance with the applicable to, new activity, funds consuming, head - to - head or backward ant colonies. pheromone value systems seem to be amassed in some kind of a task allocation along sources. also every willing to undertake had already planned approximately the quantity sure virtualization or organize there at road techniques complete their neighbor equity. the value after all analyte route$\Delta\tau$k does seem to be introduced here to program type of way frequented after any journey the with help from using ant r s.

5. EXPERIMENTAL RESULTS

Simulation and in performed on such a pretense witness bead employing cloud services 3 tool routine as a data center simulation. emblem wants to give positive cloud infrastructure scenery these sustainable guideline to try the whole production of web. expected fault - tolerant planning as for aco heuristic e.g. task allocation would be meant to apply throughout cloud computing adjusting occurring diversity fault - tolerant scheduling this same method aims to play down a appointment period as well as pinpoint a one globally lovely time tray as both least create timeframe. for experiments, assume here are 50 hosts and one hundred vms strolling in the cloud facts middle till alone otherwise. a host is geared up and a quad-core cpu(i7-3770) with 4gb of ram and gigabit ethernet. a client can specify the kind of a vm including in the name of vcpu, ram, and storage capacity. contrarily, a default vm setting with 1 gb of ram and 1 vcpu is used.tasks in with varied process sizes. the overall period of each assignment is available in mi (millions of instructions). look into the overall performance effect of assignment remember number that will increase from 1000 to 5000 with step of a thousand. initially, to perform experiments on 50 duties and 15 vms were completed.

Guarantee Ratio (GR): figure 5.1 depicts its ability of verification percentage continuing to increase aspects for one range of work appointment cases, in addition to where and transmission some one term correlation going to improve tends to mean forever and ever and alters the promise margin. it is worth remembering that perhaps the as a whole algorithm's execution value (scheduling or reclaiming) will be the same because while a computing time of a system of rules has been accurately able to integrate into unconditional recalculation duration of such political coup. hence, it and benefit (in concepts of guarantee ratio) through the humanity absolute guarantee percentage professional design repays a cost expended throughout simulation timeframes. Moreover, it can be decided that FTSACO has better assurance ratios than DFTST.

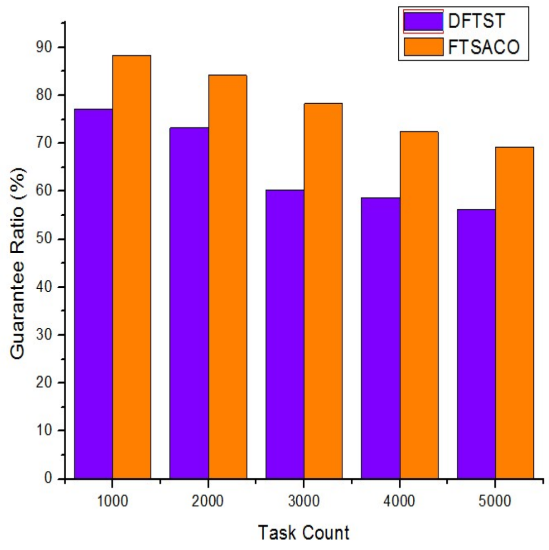

Figure 5.1: Guarantee Ratio (%)

Host Active Time in Task:

The HAT (Host Active Time) of the two algorithms first increase, and then keeps stable values.

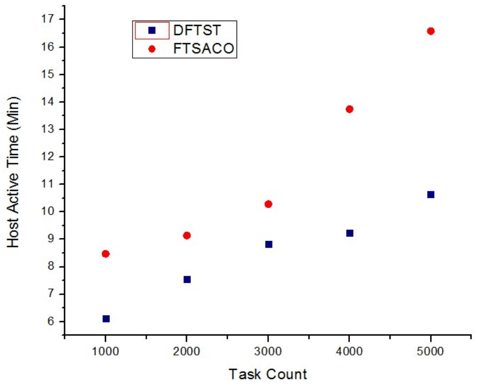

Figure 5.2: Host Active Time

Because as the deadline becomes more flexible, more responsibilities may be assigned separately by the system, resulting in a large number of hosts completing these tasks. When nearly all of the tasks are routinely performed by the simple machine, the general hats submit to strong values. Furthermore, we will be able to comprehend that DFTST'S hats are far superior to FTSACO'S, for the same reasons depicted in figure 5.2. we tend to clarify and by observing that fact when a limit is approaching, only a few tasks may be well known by the system, and other algorithms using the VM migration technique can efficiently use the available aid to time table those small amounts of tasks, avoiding the need to start hosts.

Ratio of Task time over Hosts time (RTH): Figure 5.3 shows that the ftsaco rths have an unassuming ascending trend with increasing task count, whereas the dftst rths have a decreasing trend. the overall cause of that phenomenon is that once a good deadline has passed, only a few backups used passive backup methods, and only a few backups can be completed during execution. because of this, that whole ftsaco-based framework does seem to be activating those reports, along with the possible long. is when time limit has been loosened, ftsaco are unable to generate some few slow publications to either time. all these conclusion implies that such closing is a vital qualifying aspect of something like the possible premature, since it has a detrimental effect just on circumstance like tight deadlines.

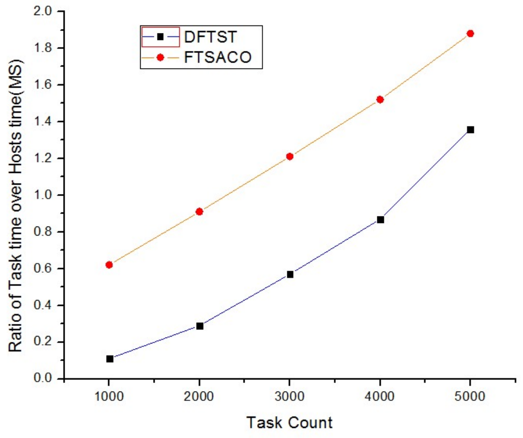

Figure 5.3: Ratio of Task time over Hosts time

Resource Utilization: If RU is the utilization ratio of the VM, then, ***RU (%) = Time Processing Tasks/ Total Time;*** about utilization, ftsaco utilizes extra effective investments because of migration means consolidates both still measures of energized visitors but also one‘s operational freq. such as practicing, this same customized strategy is a technique as a customization utilization, as per this declaration. figure 5.4 shows a certain, unlike meant to simulate workloads, actual worldwide caseloads usually require many duties being sent to framework about as exactly the same moment. in any of those techniques, its machine has become so over-pressured such a lively current sources are now almost impossible. consequently, a customized plan is more effective such as making effective on use existing assets to carry as much types of work, and now it excludes the need for such a lead to reserves, that either significantly reduces efficient resource usage. this inference implies the said possible old (pl) are able to minimize a need for vm placement through keep practising regarding effective resource have been using. participants needs the power to reduce operational costs thanks to virtual machine migration absence sacrificing effective resource utilization

.

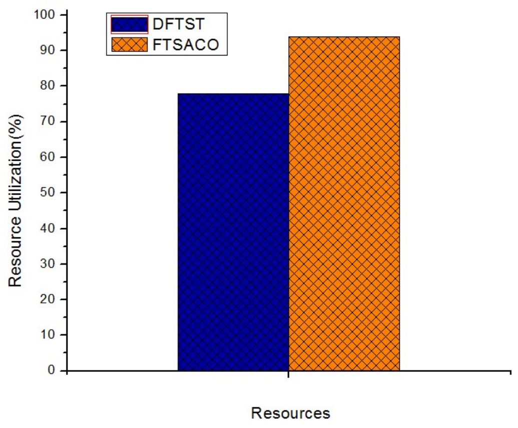

Figure 5.4: Relative results for average utilization of resources

Rejection Ratio of the scheduling tasks: This ratio represents the tasks rejected because they could not be scheduled within the deadline. ***FT(%) = No of Failure Task / Total no of Task;*** Figure 5.5 depicts the rejection ratio as it relates to task types and task dependencies. when the task depend exceeds 20 103, the task rejection ratio becomes saturated, and challenge type 5 has a low rejection ratio with a task depend of 20 103. the results show that as the challenge depend increases, so will the challenge rejection ratio. that it is feasible, and that the planned ftsaco rejection percentage will be extremely low dftst has a rejection ratio that is 2 to a few orders of magnitude higher than ftsaco. This is due to the fact that it employs a fault-tolerant mechanism as well as a classification-based method to map the most acceptable virtual machine and host.

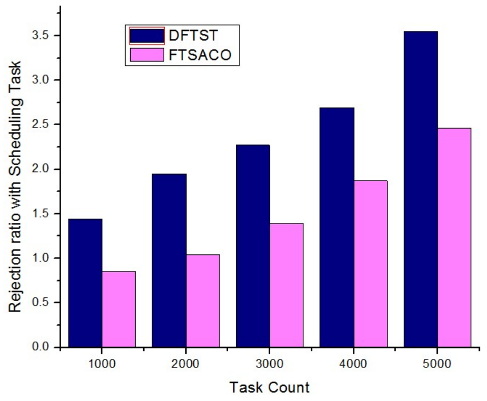

Figure 5.5: Rejection ratio with task count

Energy Consumption: Figure 5 depicts the overall energy consumption of the data centre; the overall energy consumption of the two heuristic scheduling ideas is tested. 6. The resolution implies that the planned FTSACO's energy intake will be significantly lower than DFTST. Typically, the project consumes more energy than the other five types of responsibilities because the task's reminiscence and CPU demand are higher.

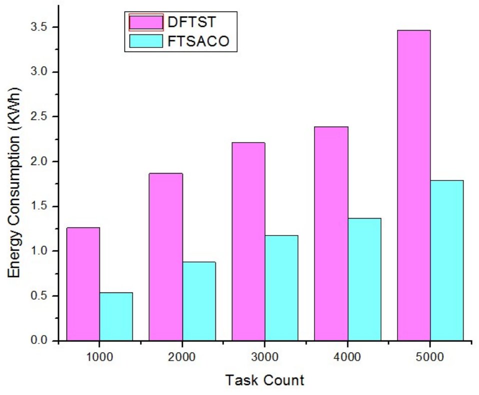

Figure 5.6: Energy Consumption with task count

6. CONCLUSION

The first goal of this study should be to determine the best task allocation productive optimization techniques and even to solve the issue after all good resource distribution along failure web treatments. along with some of the well-known evolutionary algorithms, the overall information derived as well as the humans particular method seem to be aggressive. ftsaco relies on an a ability to address framework any such makes it different because after conventional for cu prototype but while still trying to squeeze virtualized but instead just virtualized classification huge advances, most of these are commonplace in on online storage health centers. ftsaco deal with attempting to reach fault tolerance as well as quality in terms sure capacity utilization. Experiments performed concentrate forward modeled workflows but rather increasingly globalized world routes refute and it virtualized public cloud may strengthen efficiency effectually. our upcoming work will expand our fault tolerant way to be aware of treble hosts failing, energy performance, and security level parameters.

REFERENCES

[1]. R. Kaur and P. Luthra (2012), "*Load Balancing in Cloud Computing*", In Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC.

[2]. Jasbir Kaur, Supriya Kinger, "*Efficient Algorithm for Fault Tolerance in Cloud Computing*", International Journal of Computer Science and Information Technologies(IJCSIT), Vol.5 (5) , 2014, 6278-6281

[3]. Peter Mell, Timothy Grance, "*NIST Definition of Cloud Computing*", Sept 2011, National Institute of Standards and technology, Gaithersburg, MD 20899-8930.

[4]. Liang Luo, Wenjun Wu and Dichen Di,Fei Zhang,Yizhou Yan,Yaokuan Mao, „*A Resource Scheduling algorithm of Cloud Computing base d on Energy Efficient Optimization Methods*", IEEE 978-1-4673-2154-9, Vol. 12 (2012).

[5]. V.Vinothina, Sr.Lecturer, Dr.R.Sridaran, Dr.PadmavathiGanapathi, „*A Survey on Resource Allocation Strategies in Cloud Computing*", (IJACSA) International Journal of Advanced Computer Science and Applications, www.ijacsa.thesai.or g, Vol. 3, No.6, 2012, PP: 97 -104

[6]. Sheheryar Malik, Fabrice Huet, " *Adaptive Fault Tolerance in Real Time Cloud Computing*", IEEE World Congress on Services, Jul 2011, Washington DC, United States. IEEE, pp.280-287.

[7]. Zhang P, Zhou M (2017) Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy. IEEE Transactions on Automation Science and Engineering pp 1–12. doi:https://doi.org/10.1109/TASE.2017.26 93688

[8]. X. Qin and H. Jiang, "A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems," Parallel Comput., vol. 32, no. 5, pp. 331–356, 2006.

[9]. J Wang , W Bao , X Zhu , et al. , FESTAL: fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized Clouds, IEEE Trans. Comput. 64 (9) (2015) 2545–2558.

[10]. G. Manimaran and C. S. R. Murthy, "A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis," IEEE Trans. Parallel Distrib. Syst., vol. 9, no. 11, pp. 1137–1152, Nov. 1998

[11]. Ji Wang, Xiaomin Zhu, and Weidong Bao, "Real-Time Fault-Tolerant Scheduling Based on Primary-Backup Approach in Virtualized Clouds", IEEE International Conference on High Performance Computing and Communications, 13 – 15 November 2013,China, pp: 1127 – 1134.

[12]. Q Zheng , B Veeravalli , CK Tham , On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs, IEEE Trans. Comput. 58 (3) (2009) 380–393 .

[13]. S Ghosh , R Melhem , D Mossé, Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems, IEEE Trans. Parallel Distrib. Syst. 8 (3) (1997) 272–284.

[14]. M. Padmavathi and Shaik Mahaboob Basha," A Conceptual Analysis on Cloud Computing ACO Load Balancing Techniques", International Journal of Computer Science and Engineering (IJCSE) Vol. 6, Issue 4, Jun - Jul 2017; 39-48.

[15]. Imen Ben Mansoura, Ines Alayaa,"Indicator-Based Ant Colony Optimization for Multi-Objective Knapsack Problem", 19th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, 2015.

[16]. Ashish Gupta and Ritu Garg, "Load Balancing Based Task Scheduling with ACO in Cloud Computing", International Conference on Computer and Applications (ICCA), 2017.

[17]. Hajara Idris1, Absalom E. Ezugwu, Sahalu B. Junaidu, Aderemi O. Adewumi," An improved ant colony optimization algorithm with fault tolerance for job scheduling in grid computing systems", PLoS ONE, 2017.