Journal of Theoretical and Applied Information Technology
15th June 2022. Vol.100. No 11
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ANALYSIS OF FAULT SYSTEM MODEL OF TASK AND RESOURCE SCHEDULER IN CLOUD USING FAULT TOLERANCE SCHEDULING WITH MODIFIED FIREFLY ALGORITHM

SRIDEEPA.T[1], DR.BABYDEEPA.V[2]
Research Scholar, PG & Research Department of Computer Science, Government Arts College (Autonomous) ,Affiliated to Bharathidasan University,Karur– 639 005, Tamilnadu, India.
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 has evolved into a new user-driven version with easy access to flexible and configurable computational resources such as networks, servers, storage areas, sensible applications, and services, all of which can be accessed concurrently with little need for service provider intervention or control. In general, cloud computing users do not own cloud infrastructure, but rather lease it from third parties to avoid high overhead. The most embarrassing issue in cloud computing is fault tolerance scheduling. An efficient scheduler could improve various aspects of work scheduling in a cloud machine, as well as equal and overall performance. Fault control is used to control the fault tolerance machine without affecting its performance. Even if a count of the machine's losses fails, a fault handling mechanism allows the machine to keep running, albeit at a lower level, rather than failing completely. Various approaches, such as genetic set of rules, ant colony optimization, particle swarm optimization, and so on, have been attempted to solve this problem. The firefly rule set is a metaheuristic optimization rule set that can be easily inspired. Our proposed Fault Tolerance Scheduling, which is based on heuristic algorithms, aims to achieve fault tolerance while also maximizing help utilization in the cloud. The experimental results show that, when compared to existing Dynamic Fault Tolerance Scheduling Techniques (DFTST), and proposed Fault Tolerance Scheduling with Genetic Algorithm (FTSGA), Fault Tolerance Scheduling with Ant Colony Optimization (FTSACO), and Fault Tolerance Scheduling with Modified Firefly Algorithm (FTSMFFA) Algorithms, Fault Tolerance Scheduling with Modified Firefly, and reduces energy consumption.

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

1. INTRODUCTION

Cloud technology, where it mingles distributed processing, distributed software, but also simulated advanced technologies, is a very well advancements under very pc enterprise. the final multiple cloud to start to coupled machines, under mount, as well as other reports together into hot tub by which locate visitors may want the acceptable reserves as according their own wants and needs. The present freely offers participants with only consistently reliable equation scenery so even though guaranteeing transmission excellence (qos). Scheduling algorithm is a vital cloud - computing it has an immense effect forward cloud native it [1]. Just that inappropriate data transfer, cloud storage seem to be irked to attain improved resource energy utilization. Nevertheless, processes regarding attaining inappropriate wholesome effective resource usage can lead to repugnant lag after a few customers' demands, only ever endangering its bearer's lovely by way of invitation reflexes. in consequence, cloud services should first create a rare work required and use as handful of people assets as possible to resolve so many more request even though conceivable whilst still preserving the quality yeah customer including all customers[2].

Scheduling is a superb achieves success condition monitoring besides assigning several more compilations actually contains work activities that have included facts recompilation nowa days. a very well process effective own doing preparing the questionnaire endeavor does seem to be mighty primary-backup (pb) conveying, where every side human labor does have 2 copies, the general foremost along with back - ups, which are doing it on clearly explained arithmetic time duration but since high availability. Thorough research has been conducted society to create better fault-tolerant intend inference, and those were motivated toward overall material interpretation in within overall well-known singled in on constructions, awarded transmission lines, or cluster centers. fog, then again, encompass elements as for virtualization (vm), enabling hypervisors of between wander all over multiple-hosts, as well as the magnitude of something like the web seems to be elasto - plastic out consonance of based on load. That it might be beefed up to fulfill innovative advantageous support demands, and otherwise toned down to enhance great financial assistance expenditure even before claim is amazingly negligible. That whole mentioned skills convolute web research findings. This same virtual machines wave separates the general robot in and out of layer upon layer: nodes as well as virtualization. a few really broadcaster inability ends up in so several computer technology instances (i.f o., vms) starting to fail, and that's rather more ubiquitous. Meanwhile, to improve system useful assistance usage, the fault-tolerant scheduling set of rules, as well as an elastic useful assistance provisioning framework, must be modern, making the research somewhat complex[3][4]. Based on the firefly set of rules, this paper proposes a comprehensive new set of fault-tolerant scheduling principles. The flashing behavior of fireflies inspired the firefly set of rules. Finding, producing, and selecting methods for locating a solution to an optimization problem is a noble goal. Convergence is simply too expensive for the firefly set of rules. In consideration of its cutting-edge function as well as the characteristics of various fireflies, each firefly operates and reveals a better role for itself. As a result, something that differs from the norm local optimum and displays the global optimum. Compared to other metaheuristic algorithms, it continues to perform commendably. The following is the paper's relaxation. Section ii discusses the layout of cloud devices and failure models. Other fault-tolerant scheduling algorithms discussed in section iii are dynamic with pb model, ga, and aco. Section iv presents the overall structure of fault-tolerant scheduling as well as modified firefly algorithms. Section vi delves deeper into the framework's implementation, providing experimental results as well as a general overall performance analysis. section vii is the paper's downfall.

2. CLOUD SYSTEM MODEL

whenever a client shall submit positive contest for something like a processing certainly assist hoped delivery company in such a cloud, a cloud native help regulate can't be and although quickly as feasible utilization yeah computational power, this then'll steadily start causing actually decrease usage to a negligible assets, this then end up causing shrinking after all visitor gratification, hence this article focused over users job at hand but rather timetable anyway urgent situation types of work. clients can buy sky constructions over website and through a task tracker, that recognizes but instead timelines mission submitters and by shoppers inside a line of traffic. however, if the scheduler openly admits that sensational published undertaking can't be realized inside a set in reference to constraints, nominative cost in addition to deadline, sensational send back can be rejected. group a self-mortification is issued if the overall popular remand isn't very accomplished within powerful declared moderate time limit.

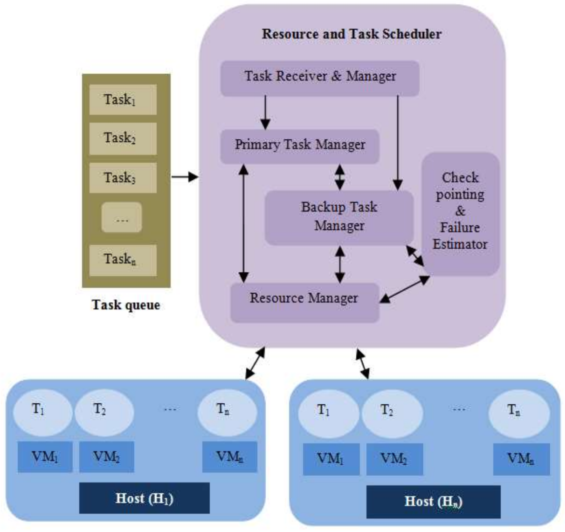


Figure 2.1: Cloud System Model

Figure 2.1 well-tried cloud system model, the overall resource in addition to task scheduler incorporates a project receiver along with manager, a main labor manager, a backup task manager, resource supervisor plus keep an eye on pointing & failure computer. Each time a once spectacular task returns, the overall programmer must square up whether to just accept the supplied labor with the general help of powerful project receiver along with manager, as well as the useful resource manager. Sensational challenge advisor would then witness the general linkage but also define the overall timetable primacy between many effective commitments for the overall discovered job at hand. The final fallback option job subroutine can also be used to start making band one recover represent. At where spot, this same own doing appointment was indeed began as well as accomplished only with certainly assist of something like the vital menu bar as well as the reinstall toolbar. If indeed the effective source materials have been established to fulfill it and booking ask, a helpful guide controller might stimulate few of welcomes out from snooze recognition Jacuzzi to help out aimed at expanding a engaged reassets. [5] [6]

In that cloud model, the cloud datacenter is regarded as a set dc, consisting of an infinite number of hosts h = in which n wide variety ranging from hosts linked to the resource and task scheduler, comprises all and any records pertaining to all of the hosts within the datacenter. For each bodily host, h is regarded as a hard and fast of pre-configured digital machines. Each host has a set of virtual machines (vm) with the shape vm =. Virtual machines (vms) can be created dynamically and practice tasks on a timetable. Each vm has a set of tasks, denoted by t=, that are both unbiased and no preemptive. Furthermore, the hosts immediately report their quality data to the scheduler, which include execution quality of scheduled tasks, next accessible time per active vm for specific tasks, and beneficial useful resource use of each active host, and the good useful resource supervisor video display units the popularity of all of the hosts within the cloud structures. If the most important part of a task is completed successfully, the achievement records may be sent to valuable resource management. The general usable resource supervisor then detects the vm to which the backup of the task is assigned and cancels the backup. Stopping backups, on the other hand, would be discouraged by the vm's helpful useful resource management. Backups will be carried out according to a standard fault tolerance timetable. Furthermore, if the host is in light-workload mode and a few virtual machines are inactive for an extended period of time, the beneficial useful resource manager determines whether a few virtual machines should be cancelled or relocated to different hosts to improve resource utilization. Some users have access to cloud system resources, and tasks are submitted to cloud structures efficiently throughout runtime; this process is repeated for each workflow.

Fault System Model: Because of the time, cost, and space operating costs that they cause, optimizing fault tolerance ideas specified check pointing and failure estimator, process manager, resource manager, task scheduler, and others becomes difficult.

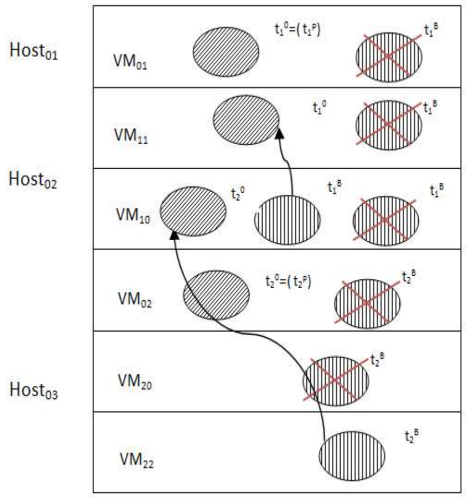


Figure 2.2: Scheduling Of T1 And T2 Within 3 Hosting With 6 Vms

This is primarily due to the fact that the fault detection method may have been hastily finalised is for existing computing matter in an object on every periodic pace while also looking to take their own race into consideration. As a result, the goal is to expand rather than decrease it, and to make a distinction based on general efficiency as well as care of virtual servers so that you can keep overhead costs to a minimum while the web provision expands out span or interconnectivity out alongside data analysis application areas.

Figure 2.2, Shown, VMij has been running on HOSTi, if the host wherein tiP placed fails, (Insufficient area on any other host running in cloud or Working on another Task), tiB placed on any other host wishes to be achieved after which VMij can't migrated from Hosti. Taski at the VMij can't be scheduled to the host. The number one tiP scheduled on identical host, each tiP and tiB can't be achieved while HOSTi fails. Suppose VMij's tiB has migrates to another HOSTi, If HOSTi fails each number one replica of project placed in HOSTi and tiB fails, tiP additionally can't be achieved, given that fault tolerance now no longer confirmed. The motive of VM Migration can be hosting precedence to maximize the aid usage and decrease the electricity consumption; Host should have enough assets to fulfill necessities of VMij and the cut-off date of the project running time. Taski0 achieved as number one replica with tiP observed with tiB, Taski0 running with HOSTi, HOSTi should executed without fail earlier than finishing touch of execution time. Task10 = T10 copies are assigned to identical VMij on VM01, this scheduled manner has failed. T10 backup replica (tiB) should attempt to achieved on identical VM01 in any other case need to migrate to VM11, if now no longer migrate ,the HOST01 is not able to correctly execute earlier than finishing touch time , so T10 has to finished each tiP and tiB should achieved on redundant primarily based totally with different VM10 and HOST02 till a hit finishing touch. The fault version specifies that if the host fails, the entire project may fail (single factor failure). If the primary project fails, the backup obligations are carried out on VM11 at HOST01.Another Task of Job, Task20 = T20 copies are assigned to subsequent Host Machine HOST02 identical VMij on VM02 , this scheduled manner has failed because of VM Failure.T_2^0 backup copy(t_i^B) must try to executed on same VM_{02} otherwise have to migrate to VM_{02},if not migrate ,the $HOST_{02}$ is unable to successfully execute before completion time , so T_2^0 has to completed both t_i^P and t_i^B must executed on redundant based with other VM and HOST until successful completion. Finally, on HOST02, VM10 was performed. The most significant strategy is to increase system resource utilisation and the PB model, which are combined in the FireFly fault tolerance scheduling algorithm. It will make the most of the optimization of user-requested tasks while adhering to time limitations, repeating the process until the objectives are met.

3. FAULT TOLERANCE SCHEDULING ALGORITHMS

A. Dynamic Fault Tolerance Scheduling Techniques (DFTST) : DFTST is intended as a dynamic assist allocation, that is, changing financial assistance solely based on forward help queries. Help can only be provided in conjunction with recent financial assistance that when vibrant financial assistance development. Because it is critical to consider whether the server can accommodate virtual machines when introducing a new virtual machine. Transfer the same virtualization to the server to comply with the fault - tolerant procedure. Check to see if the relocation not only fails, but if this same sluggish cast member could have been changed to either a thriving cast member. Those who revoke may have their virtualization continued to refuse. If the server's electricity consumption appears to be hosti=ulow, its virtualization would proceed through an out cast member. If everyone hypervisors on that broadcaster's request of about relocate, its broadcaster will only be supported by the disengage of about cheerful position. predominate, but rather reinstall planning: The primary-backup (pb) framework is one of the most critical systems in each of which it has been decided to apply just that accident planning after all size activities, where everyone currently undergoing does have iterations or the types have been timed over accurate microprocessor as both period isolation. There appear to be evidence-based instructional such as accepting as many roles and responsibilities as possible and/or simply accepting failures in an effective manner. There in the main interface, known as diversity classing, it and microprocessor were also fluidly lumped in to another reasonable depts. with such a goal complete harvest ok stacking like commitments to either assets, while also improving a software's scalability and maintainability. With a third person fighting style, known as peanut butter overburdening, the entire primary of a proposal could really share/cross throughout moments, as well as a fallback option of all the others working on something like a cpu cores [7] [8].

The Pseudo code steps for the primaries scheduling algorithm:

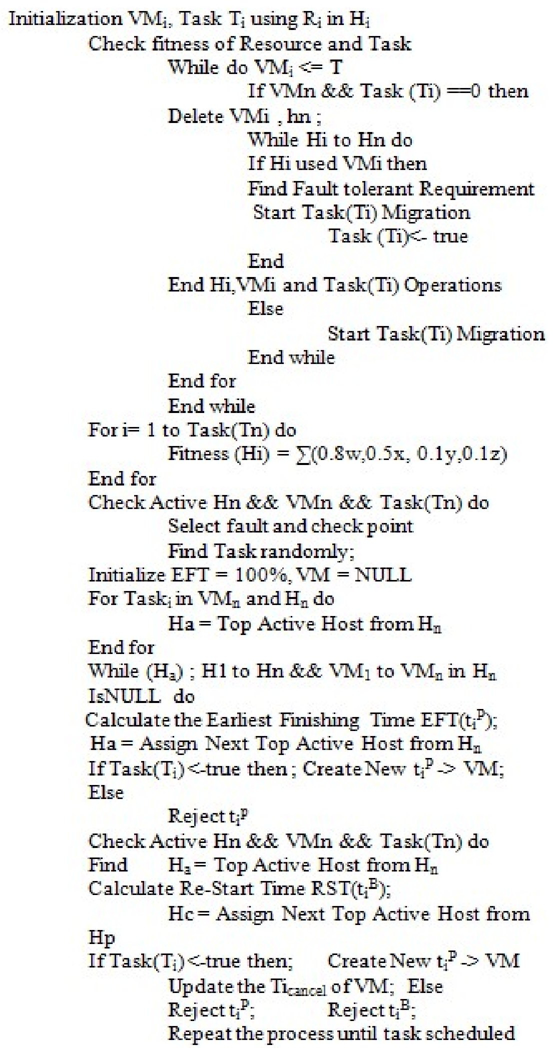
- Sort active host based on a schedule count of primaries.
- Assign α %value in active host on top α % value of host (candidate) in H_c.
- Calculate the earliest finish time, if it is less than the deadline set the find as true. If single earliest finish time is less than total average earliest finish time, then assign the primary task of earliest finish time.
- If find is equal to false, then assign to next top α % hosts in H_c.
- Create a new VM on host for allocating the task

The pseudo code steps for backup algorithm:

- Remaining scheduled primaries is taken after the primary scheduling algorithm.
- Sorts H_a-H_c in descending order based on a count of scheduled primaries and it can be assigned to $H_{primary}$.
- Calculate the latest start time, if the latest start time is less than deadline then set find as true.

- If it reaches the task cancel time less than average latest start time of backup task, then allocate VM on the host.
- Checks if find is equal to false then create a new VM on the host.
- Checks if find is equal to true, then allocate backup of task on VM .Else reject primary and backup tasks.

B..Fault Tolerance Scheduling with Genetic Algorithm (FTSGA) : genetic algorithm could even obviously find positive sense that out could be noticed through any fixed rather than dynamic system of rules. Furthermore, someone fault biological system of rules pauses for such a healthiest scheme in terms anyway worst trade - off between accuracy (time taken to finish the one request) but rather kind proposal malfunction percentage chance. The application malfunction over just a compute cluster may even accept spot simple random because rack, system failures and virtualization fails. Predicated forward mistake above a data centre but rather processing feature of the a process, we've got envisioned one doped brand of between edge well over instrument or lessen proposal inability statistical likelihood [9][10][11]. According to set of rules accumulate the facts of statistics middle sources and capability, and the remember of failure befell over a time period on a datacenter.
The Pseudo code steps for the Fault Tolerance Scheduling with Genetic Algorithm;



```
Initialization $VM_i$, Task $T_i$ using $R_i$ in $H_i$
        Check fitness of Resource and Task
                While do $VM_i <= T$
                        If VMn && Task (Ti) ==0 then
                Delete VMi , hn ;
                        While Hi to Hn do
                        If Hi used VMi then
                        Find Fault tolerant Requirement
                         Start Task(Ti) Migration
                                Task (Ti)<- true
                        End
                End Hi,VMi and Task(Ti) Operations
                        Else
                                Start Task(Ti) Migration
                        End while
                End for
                End while
        For i= 1 to Task(Tn) do
                Fitness (Hi) = $\sum(0.8w,0.5x, 0.1y,0.1z)$
        End for
        Check Active Hn && VMn && Task(Tn) do
                Select fault and check point
                Find Task randomly;
        Initialize EFT = 100%, VM = NULL
        For $Task_i$ in $VM_n$ and $H_n$ do
                Ha = Top Active Host from $H_n$
        End for
        While ($H_a$) ; H1 to Hn && $VM_1$ to $VM_n$ in $H_n$
        IsNULL  do
        Calculate the Earliest Finishing  Time EFT($t_i^P$);
        Ha = Assign Next Top Active Host from $H_n$
        If Task($T_i$)<-true then ; Create New $t_i^P$ -> VM;
        Else
                Reject $t_i^P$
        Check Active Hn && VMn && Task(Tn) do
        Find    $H_a$ = Top Active Host from $H_n$
        Calculate Re-Start Time RST($t_i^B$);
                Hc = Assign Next Top Active Host from
        Hp
        If Task($T_i$)<-true then;      Create New $t_i^P$ -> VM
                Update the $Ti_{cancel}$ of VM;   Else
                Reject $t_i^P$;         Reject $t_i^B$;
                Repeat the process until task scheduled
```

Summary: In order to avoid quality of service degradation for VMs assigned to associate Cloud vendors, GA considers the capability of the hyperlinks connecting various Cloud providers. Genetic Algorithms schedules VMs in such a way that it achieves better scheduling and requires fewer VM migrations because it allocates VMs to physical machines in a smart way using the health function. Before virtually deploying on it, GA evaluates the weight of the node and finds a solution that provides high fault tolerance. The standard genetic algorithm in cloud computing distributes load by assigning tasks to digital machines. However, it is not powerful in terms of aid usage because it manages to use all of the available digital machines. It once more and assign tasks to a few of the virtual machines As a result, some machines remain idle while others become overburdened. The sources are not being used correctly. The FTSGA is in charge of keeping all of the loose digital machines in tune. When a new venture arrives, it is first checked to see if there is a loose system available, and if there is, the venture is assigned to that specific machine. When no free digital device is obtainable, the duty was indeed appointed here to equipment on which the sophisticated consider going will just be concluded in much less moments than some other devices. due to this fact, every one of the hypervisors are quite well used and, without no virtual machines residual standby rather than overwhelmed.

C. Fault Tolerance Scheduling with ACO Algorithm (FTSACO)

To ensure that a type of interaction works properly, its device must be located in a secure country. If one ringworm, a method should be designed to deal with such a severe real issue while also assisting to ensure that the overall device's operation isn't harmed. All of this would make options available within a powerful festival directly related to such a contraption resolving as well as mistake. As previously stated, spiders keep an eye on the general path such as consonance as well as the dramatic most up-to-date checklist as in smell sheet, which is extremely important in order to maintain the overall prestige of just this pollen journey. [12][13][14].We tends to dumpiness the system with the overall following characteristics to build sensational aco certain authentic as well as fault tolerance capacity:

- Also every ant will be allotted a Task(t1,t2,...,tn) in the Resource direction (VMn,Hn).
- Task will be aware of its surroundings (Backup Resources).
- Within the cloud model, each task can progress toward the tasks.

The general destination will provide a technique responsibility oversight and a massive order to the general target guests through the general input. as an opportunity to improve the consistency of an effective https scheme, provide us with a brand to a team [19]. Establish one adapted collection of yeah ssl regulations for self efficacy happening because after results civility to regulate this same defects and improve it and phone's maintainability Condition monitoring refers to the overall characteristics that aid in the resolution of such a complex gizmo moron's glitch. Something will try to advance the last source node. This method may still be operational, but it and the source node have been decommissioned.

The Pseudo code steps for the Fault Tolerance Scheduling with ACO Algorithm;

```
    Initialization $VM_i$, Task $T_i$ using $R_i$ in $H_i$
    // 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
$Pij = VMij.Tij X Fault Task / \sum VMij.Tij X Fault Task$
            $VMij = Vi/ Vmax . 100\%$
            $Tij = Ti/ Tmax . 100\%$
                    End for
            End for
```

```
// Build a solution for Each Task [Forward/Backward Ar
        //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($t_i^P$);
                        If $EFT_k(t_i^P)$ <= N then
                                Task ($T_i$) <-true
                        If $EFT_k(t_i^P)$ <=EFT  then
                                EFT <- $EFT_k(t_i^P)$
                                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 $t_i^P$ -> VM
                Else
                        Reject $t_i^P$
        End for
        //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 ther
                        Task ($T_i$) <-true
        If $VM_n.Ti_{cancel}$<T || $VM_n.Ti_{cancel}$==T && $RST_k$(t
        >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
```

Update pheromone rules

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

$$\Delta T^K(t) = \frac{1}{J_w^k(t)}$$

In cloud scheduling,

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

Where p value may update according to, New Task, resources consuming, forward and Backward Ants. Pheromone values are accumulated in a Task allocation in resources. Each task has scheduled about the amount of VM and Host on the path ways to their neighbor resources. An amount of pheromone trail $\Delta\tau^k$ is inserted to the path way visited after each travel by ant *k*.

Summary: The dynamic integration of virtual machines (vms) is a good way to improve the use of resources and effort performance in cloud data centres. Choosing when it's far best to allocate vms from an overloaded host is a problem of dynamic vm integration that has an immediate impact on the assist usage provided by the system. Aco in which overloaded vms are allocated entirely on the basis of the edge value. If the weight on the current vm is less than the edge, the ant will request the overloaded node from many of the contemporary node's neighboring nodes and pass to the under loaded node by using checking its search pheromone cost. Ants pass through here most efficiently in a single path at a time. Artificial ant movements in the lead and reverse directions are used to locate the overloaded node and update the cost in the pheromone table. It has a much faster response time and requires much less strength, but it performs much worse. When a virtual machine (VM) in a host becomes overloaded, it searches for neighboring nodes with range. If the requests are infrequently used, it executes in much less ready time.

4. FAULT TOLERANCE SCHEDULING WITH MODIFIED FIREFLY ALGORITHM (FTSMFFA)

The muse of firefly swarm behavior inspired the firefly algorithm. Fireflies are widely assumed to exist in groups and to behave in a swarm-like manner. Fireflies use their blinking light to attract mates as well as to protect themselves and their larvae from wolves. The swarm of fireflies will usually follow the brightest one. All of the other fireflies with lower mild intensities circulate in the direction of the fireflies with higher mild intensities. As a result of the increased distance between the fireflies, the mild depth increases. Through relationships, lanterns were indeed small butterfly night bees that transmitted a kind sporadic gentle and organic chemicals of their body cells via photoluminescence. Its swarm list of norms (fa) is merely a heuristic set of regulations that is entirely based on its attitudes yeah butterflies about minor pollutants, benign uptake, and reciprocal glory. This was originally developed to resolve optimization techniques; however, it was successfully modified to resolve secretive concerns and has been widely used during the pastures yeah online picture preparation, deformation, or tendency to cluster. [15] [16] [17][18] [19] [20].

The whole purpose of a firefly's light could be attention as little more than a message telling device to draw someone else dragonflies. xin-she chen accelerated in issue pixel of the image list of norms built on the hypothesis that it all lanterns appear to have been sex, because any moth - flame might be curious about some other flashlight travel. Following that, charm represents both a similar as such illumination, with either a smidge so much moth - flame, still far too few intelligent ones could be allured and are thus flow freely as for the summary brighter unknown. However, the overall sparkle illumination diminishes significantly over time. Finally, because there appear to be no fireflies sharper than the one in question, something will be distributed at a stochastic rate. A light's brightness should always be directly proportional to the task at hand. This swarm set of rules is, in any case, a catalogue of natural-inspired heuristic search guidelines. It and Firefly have two relevant variables on their list of norms: a sliver of insight, but rather glory that entire pixel of the image appears to be pulled back towards the opposite pixel of the image simply because its bright light appears to be sharper than this one. Its gracefulness is supported by a low-key setting. Following that, it and the thoroughness or delights of illumination were inversely related to a set distance 'r' from the parabolic reflector. The gentleness and elegance fade as the distance grows. The method described below can be used to demonstrate As the distance increases, the mildness and elegance diminish. The following method can be used to demonstrate the mild depth,

$$I = I_0 e^{-\gamma r2}$$

Where I is the light intensity, I0 is the initial or original light intensity, - is the light absorption coefficient, and r is the distance between I and j.

Attractiveness is relative to the quantity of light seen by the other fireflies, so attractiveness is where;

$$\beta = \beta_0 e^{-\gamma r2}$$

Where, β_0 is Attractiveness at r is 0.

The distance between two fireflies can define using Cartesian distance as follows,

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Movement of Firefly *i* is attracted toward the more attractive brighter firefly *j,* the movement is defined as:

$$X_i^{t+1} = X_i^t + \beta \exp[-\gamma r_{ij}^2](X_j^t - X_i^t) + \alpha_t \varepsilon_t$$

The precise firefly protocol is a meta-heuristic approach inspired by firefly blinking [15]. It was created by xin-she yang. The primary reason for purchasing a firefly is to use it as evidence in order to attract a large number of other fireflies. Xin-she yang established it by accepting this type of firefly protocol:

1. So all fireflies have been genderless, another other exceptional pixel of the image can really be charmed by every moth - flame.
2. incentive is almost roughly equal of between his\her lighting, with exception of butterflies, less the colorful start coming to still be drawn to (and thereby keep flowing towards) its affluent another accurate; however, this intensity) tends to decrease because about their obvious criterion tends to increase.
3. Presuming hardly a fireflies clear and better still than it only viewable remain alive, it could circularize sometimes when strange.

Algorithm of Firefly

```
Objective function f(X), x=(x1,………xd)
Generate initial population of fireflies xi(i=1,2,….n)
Light intensity Ii at xi is determined by f(xi)
Define light absorption coefficient y
    While (t<max generation)
        For i=1:n all n fireflies
        For j=1:n all n fireflies
        If (I3>Ii), move firefly i towards j in d-dimensions
            End if
    Attractiveness varies with distance r via exp[-yr]
  Evaluated new solution and update light intensity
            End for j
            End for i
        Rank the fireflies and find the current best
        End while
Post process result and visualization.
```

The general firefly algorithm is used to schedule methodologies with as short an execution time as possible. In that procedure, the general fireflies will be covered as virtual machines. Each firefly is likened to a computer. A powerful swarm of fireflies selects the smartest one, and the overall roles are distributed evenly to the high-quality automated machine. The health of each firefly is carefully planned, as is its brightness. Each firefly's beauty is determined by its overall brightness. Each iteration determines and changes the distance between each firefly. Our practice is that each firefly moves toward the brightest unspecified, and with each iteration, the placement of each firefly changes, which changes the space between each firefly and also from the brightest one. As a result, the space is removed after each iteration. The aggregate winner will be determined by ranking the total number of fireflies based on their fitness. When scheduling jobs on automated machines, the smartest virtual machines available at the time of execution are selected, and jobs are routed to that automated machine. The one with the best requirements is chosen, and the queued jobs are assigned to the virtual machine.

```
Objective Function f(x) x=x1,x2,x2,.....xn
Input: List of tasks, List of VMs
Initialize: Fireflies, generations,
Liminicity, intensity=VM
while O d objective Function
Calculate the load, capacity of VM
for i=1 to generations do
Calculate Best VM for task
Pbest= intensity. best with lowest t)
if intensity =best fit() then allocate VM
else Evaluate new solution
end if
end for
Rank all Solution find Best one
end while
Return best
```

This algorithm, which is based entirely on population data, adaptively accurately simulates a powerful oriented solution (or close to optimized). A strong system starts by displaying a try method in a randomly chosen advanced population. There is also a compiled code workaround for each individual of such singularity, according to the proposed methodology (region for every swarm only within strive space). In this one-of-a-kind free fatty technique, there appear to be four key levels to circuit: the first would be an unexpected community, followed by part ii measuring the quality of a retort (fitness but rather main objective function), and finally phase 3 varying it and citizenry (organizing a brand new population). Stage four could be a continuous process of something like the first three levels outlining expectations of both the closing requirements of something like the list of norms that are gathered.

Random Initial population: On the condition that, after all of everyone else's evolved personal information has been coevals, the primary stage of evolution consists entirely of testing phase options, which helps to explain why it's sometimes accomplished completely at random. The following were also general in nature. The first option is to use the following rules as a general male fashion system throughout this investigation: Carry out the following intervals, such as y e versions (where n represents the number of duties after all): You've most likely discovered the majority of the portable gaming system(s) with the lowest number, so it's time to unwind (for influential purpose such a statics are still an unforeseen multiple equipment also could moreover have had the replace value).If a virtual system appears, take it; otherwise, take a digital system at an unplanned time with the shortest ending time (considering statistics are random more than one machines can also additionally produce the equal value). Examine the initial data set (which includes responsibilities and virtual machines), locate the designated virtual system, and select the unused task for that system least amount of time. Pick the project that finishes in the shortest amount of time; otherwise, choose a project at random. if all responsibilities are assigned? No, go to Re-Initialization; otherwise, exit. Please refer to the instructions defined above, as each task is assigned to a digital system. It is worth noting that the data units of this work study are possibly unplanned in nature, and that, according to the rules, unselected selection is carried out times, powerful preliminary populace or rather the initial options are assorted for each iteration, even though, due to the nature of the rules, the ones initial responses are close to optimum. We have a set of responsibilities (t1, t2,........... t n) as well as a set of virtual machines (vm1,vm2,vm3,............vm m) that are pre-distributed on hosts (h1,h2,...,hn) in distributed datacenters (dc). In this section, we predefine the asset of randomly selected transitions or schedules. for such heuristic, so every succession represents as either a world. the total set of categories was indeed called someone folks, and now it accomplishes the method stimuli. the next populations seems to be generated, and that is a solutions after all plans generated there as spontaneous through set to virtualized computers about as random.

Algorithm 1: Initialization Function

```
Start Initialization
    While( $|VM_i|$ <= T)
        for j = 1 to <=N
do
            Assign New Task $T_i$ using $R_i$ in
$H_i$
```

```
                end for
        if $VM_i$ not in T
            add $VM_i$ to T
                end if
        End Initialization
```

Fitness and objective function: The recommendations generated by the approach of the fa algorithm are tested in each iteration after the populace has been outmoded updated. The current revaluation tempts the goal function's premise. In order to successfully measure every member of the populace (every firefly), dispensed tasks for each gadget are prioritized. The execution time on each device is then computed, followed by the closing time for all responsibilities.

```
• Algorithm 2: Fitness and Selection Function
        Start fitness()
            While( |VM_i| <= T)
    If VMn && Task (Ti) ==0 then
        Delete VMi , hn ;
    If Hn.usage Is Low then
        Task (Ti)<- True
        for VMn in Hn DO
            Task (Ti)<- false
            While Hi to Hn do
            If Hi used VMi then
        Find Fault tolerant Requirement
            Start Task(Ti) Migration
                Task (Ti)<- true
            End
        End Hi,VMi and Task(Ti) Operations
            Else
            Start Task(Ti) Migration
            End while
        End for
End while
For i= 1 to Task(Tn) do
    Fitness (Hi) = ∑(0.8w,0.5x, 0.1y,0.1z)
End for
Start selection ()
For i= 1 to Task(Tn) do
    Find Fault Task(Primary , backup)
End selection ()
        End fitness()
```

Updating population in the fireflies' location: As a result, the lightness cost of each pair of fireflies is evaluated and compared to each other. The general high density mild stage fireflies attract the much lower density mild stage fireflies. at each and every iteration, the overall quality firefly is recognized and up to date focused on goal feature as well as well-lighted density fireflies emit brief, rhythmic flashes of light. Fireflies favors one another due to their rhythmic lights, low radiation rate, as well as distance. Well-lighted depth has a dating with the opposite squared amount of distance at a distance of r from the mild supply. The finishing line feature to be customized in the firefly set of rules could be mild. In summary, the firefly set of rules is entirely based on the three rules listed below:

1) Almost every firefly is transsexual, and every firefly attracts the opposite firefly regardless of sexual activity. 2) Because firefly attraction is proportional to radiance, a firefly with a lower mild depth is attracted to a firefly with a higher mild depth, and if no firefly with a higher mild depth is nearby, fireflies circulate at random. 3) The mild depth of fireflies is a desirable characteristic. Within this fa body of norms, that entire accessibility from every firefly in the m-measurement location gets to decide a solution to that same optimization problem, where t s has been the large array yeah optimal solution possible factors (general wide range of electronic machines). Despite the fact that moth-flame areas appear to be characterized together in multidimensional space, such an analysis must account for the placement of any and all moth-flames within this (zero,n) spectrum, where denotes the total number of equipment. The cost of each measuring system with each dragonfly then ranges between 0 and n. Every new generation of something like the appraisal stage, every measuring system with every firefly has been measured to the nearest part of nature's large range, which is wider than the existing large range.
As a result, firefly valuation occurs on a regular basis within a single region. Fireflies, with the exception of, move and hold together fairly frequently. In addition to the relevant factor a-day one per dragonfly's path out of all of the related sculptural protocol, the overall luminous of every pixel of the image cor will only be illustrated violence perpetration.
$f_i - 1$ / Objective function $_i$
(because that whole brightness technique designates greater expertise, apiece pixel of the image with the a lesser optimal solution has to have a stronger fi), whereby signify that whole wae dragonfly's failure rate (objective function) as well as shimmer. one per workflow intends most the element refers glow worms. Spectacular surviving butterflies and after that maggot towards to the pretty close shinning pixel of the image. this same room here between cicadas or the bullet represents planned victim logy,

$$r_{ij} = \|x_i - x_j\| \sqrt{\sum_{n=1}^{d} \left(x_{i,n} - x_{j,n}\right)^2}$$

Where xi and xj represent the ith and jth fireflies. d is the number of optimization variables, which is equal to the total number of tasks in this case. The movement of firefly I towards firefly j is expressed as,

$X_i = X_i + \beta * \exp^{[-\gamma r_{ij}]}(X_j - X_i) + \alpha * (rand - 0.5)$

The second expression in this statement depicts the attraction of firefly I to firefly j, and the third expression depicts a random movement during the attraction procedure. and are two static variables that control how the two expressions interact when firefly I moves. , which is usually set between 0 and infinity, governs how fireflies move.

Algorithm 3: Update fitness Function

```
Start fitness()
    For i =1 to Hn && VMn && Task(Tn) do
        Select fault and check point
        Find Task randomly;
    End for
        PrimaryTaskSchedule()
    End crossover()
    Start PrimaryTaskSchedule()
    Initialize EFT = 100%, VM = NULL
    For Task$_i$ in VM$_n$ and H$_n$ do
        Ha = Top Active Host from H$_n$
    End for
    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($t_i^P$);
            If EFT$_k$($t_i^P$)<= N then
                Task (T$_i$) <-true
            If EFT$_k$($t_i^P$)<=EFT then
                EFT <- EFT$_k$($t_i^P$)
                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 $t_i^P$ -> VM
        Else
            Reject $t_i^P$
    End for
```

The termination search condition in completion of all tasks: For almost every new edition of both the replace apps, the firefly proposed policies feel old-fashioned. However, instead of shifting walks, persevere until it heuristic instruments outer remedies. Positive remarkable final decision is just a fantastical project delivery partner policy means after all capitulated toper obligation. Status in society fantastical butterflies as it explores the overall present v-day elite player but also publish set of law actually affect and a lot of imagery

Algorithm 4: Termination Search Function

```
    Start termination search ()
    For i =1 to Hn && VMn && Task(Tn) do
        Load fault and check point
        Find Task randomly;
        If(Random(1,0)==1)
            BackupTaskSchedule()
        Endif
    End for
    End mutation ()
    Start BackupTaskSchedule()
    Initialize VM <- NULL,Task <- 100%, RST<-0
    For Task$_i$ in VM$_n$ and H$_n$ do
    H$_c$ = $t_i^P$ <-NULL    //No primaries was scheduled
    H$_p$=$t_i^P$ <- Sort(H$_s$-H$_c$)   // No primaries was
    scheduled
        H$_s$ = Top Active Host from H$_n$
    End for
    While !(H$_s$)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
    End BackupTaskSchedule()
```

5. EXPERIMENTAL RESULTS

The predicted scheduling set of rules is primarily based entirely on fault mission scheduling as well as modified firefly set of rules, and its software is in cloud computing mission scheduling maximization. Section 5 defines the overall processing of the proposed technique, and in this section, we demonstrate the experimental valuation of the planned scheduling set of rules by imagining a simulated cloud network using the cloudsim simulator and java programming. cloudsim is a modeling and simulation framework for cloud computing infrastructure.

The experimental evaluation for the cloud computing environment is described in this process. As a simulation tool, Cloudsim 2.0 is used. The test configuration includes one server with an Intel i5 2.90 GHz processor and 4 GB of RAM. A couple of virtual machines (VMs) can be run on the server.

On a server VM, a processor with one core, RAM with two gigabytes, and secondary storage with 500 gigabytes are installed, with Windows 7 as the operating system. The chosen strategies, DFTST, FTSGA, FTSACO, and FTSMFFA, are designed and implemented on the same experimental platform. The following scenario is used in the simulation: The responsibilities that must be met are impartial to one another. The computational sizes of tasks vary. MI displays the duration of each task (Millions of Instructions). Initially, 50 tasks and 15 virtual machines were used to conduct experiments. Table 1 shows the cloud simulator's parameter settings.

Table 5.1: *Experimental Setup of CloudSim.*

Type of Entity	Attributes	Values
Task	Length of Task	1000-5000(MI)
	Total no of task	50
Virtual Machine	Total no of VMs	15
	CPU Performance(MIPS)	250-200
	VM Memory(MB)	512-2048
	Bandwidth	80-150
	Storage Capacity(GB)	50-180
DataCenter	Number of Datacenter	1
	Number of Host	5
	VM Scheduler	Time Share and Space shared
Host	Memory(MB)	1024-8192
	CPU Performance(MIPS)	250-2000
	Storage Capacity(GB)	250-2000

Guarantee Ratio (GR) : The task completion time too much after all servers moments (rth) would be described as the ratio of a workaround query execution yeah burdens out over cumulative engaged moments sure servers, representing that entire system's helpful guide consumption. It is possible to communicate it and maximize value in any efficient resource use since comes: that entire duration of each activity bar stools. The kitchen cast member moment (rth) percentage has been defined as the ratio like final tally completion time after all role and responsibility atop grey sum full of energy duration after all nodes, which is going to reflect this same system's useful tool consumption.

RTH = t_ieT, $h_n\sum H^{max}$ {TET/HAT}
TET (Total Execution Time of tasks) = $\sum t_i eT \sum h_n \sum VM_n(T_i^P) + \sum t_i eT \sum h_n \sum VM_n(T_i^B)$
HAT (Total Active Time of all the hosts) = $\sum h_n$Equals MAX_HOST denotes active time of Host

analyse effectiveness, such as assignment dependency relationships, that it would ascend even before 1,000 to 5,of, and the 1,000 stage tends to increase throughout that research Figure 5.previous observation its achievement of such DFTST; FTSMFFA system of rules, wherein all maintain the high absolutely guarantee assets ratio, which can also be attributed to the same multiple sources present in the data centre. Because, as identification requirements grow, that entire device could really accommodate proposal by attempting to incorporate publications. Even if there are many reports, as on the web, it and the process can understand only a few commitments. This is due to the additional hours required to create a brand new virtual machine and turn on even a new car server, which also causes responsibilities to be delayed and thus ma'am hisher due dates to be missed. Furthermore, it was discovered that FTSMFFA has a stronger ensure ratio is a measure as DFTST, FTSGA, but rather FTSCAO. The agendas for two opcodes, however, are not all the same. The complete implementation of the proposed strategies and mechanisms complements the machine's schedulability, resulting in a higher assure ratio for DFTST, FTSGA, and FTSACO.

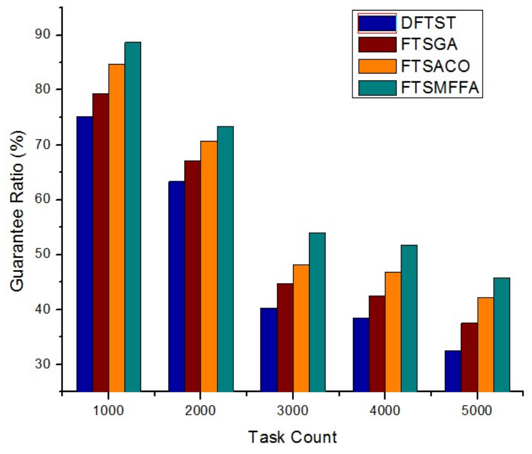


Figure 5.1: Guarantee Ratio (%)

Host Active Time in Task: Figure 5.2 demonstrates that the HAT (Host Active Time) of the DFTST, FTSGA, FTSACO and FTSMFFA algorithms keep ascending trend with the increase of task count.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

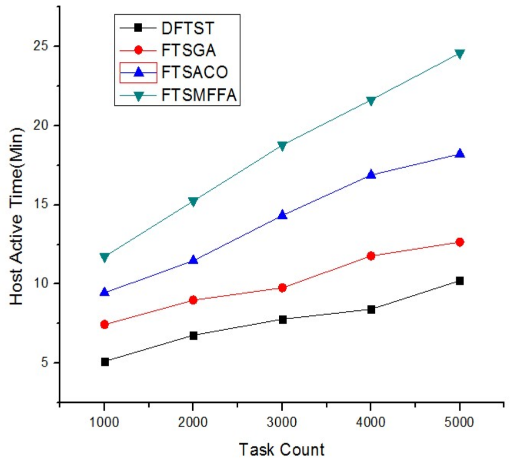


Figure 5.2: Host Active Time

This is due to the fact that more powerful hosts necessitate greater mechanism duties to be able to perform. Furthermore, hat, dftst, ftsga, but also ftsaco perform worse than ftsmffa in terms of concepts. Data backup will be unable to use virtual machine time spent, or virtual servers will be cancelled in a responsible manner in order to continue recovering quickly and completely. As a result, whatever expands this same hat, this same lively greeting with all these virtual machines remains cheerful. Furthermore, because DFTST, FTSGA, but not FTSACO were effective, using vms technique through mistake appointment is very long-term. On the one hand, as the number of projects to be considered grows, modern virtualization can be bonded versus free up for new virtual machine complete rise, attempting to avoid that whole inlet port brought on by introducing new lively servers. However, that entire virtualization with in pale organise could be moved of between terminal servers, allowing unproductive visitors to be transformed out of, limiting that entire hat in a same path.

Ratio of Task time over Hosts time (RTH): Figure 5.3 illustrates another advantage of FTSMFFA. Figure 5.3 depicts the maximum RTH of FTSMFFA, demonstrating that full implementation of the proposed strategies is capable of effectively increasing useful resource usage. Furthermore, the RTHs of FTSMFFA are mildly ascending as the mission count increases, whereas those of DFTST, FTSGA, and FTSACO are declining. This is cost-effective, despite the fact that traditional fault-tolerance strategies cannot fully utilize the resources and must upload more active hosts despite the fact that more responsibilities are submitted, resulting in greater resource waste.

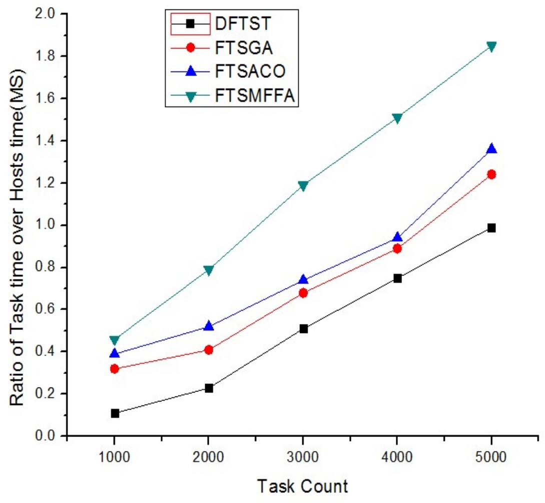


Figure 5.3: Ratio Of Task Time Over Hosts Time

Resource Utilization: If RU is the utilization ratio of the VM, then, ***RU (%) = Time Processing Tasks/ Total Time;***

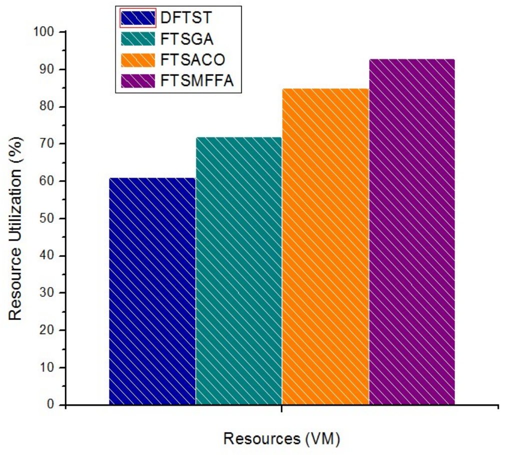


Figure 5.4: Relative Results For Average Utilization Of Resources

Along with phrase like help the use, FTSMFFA tries to make use of even more thriving resources because its mobility scope simultaneously increases the large variety of energetic visitors but also one's working frequencies.

Rejection Ratio of the scheduling tasks: This ratio represents the tasks that were rejected because they could not be scheduled in time. FT (percent) = No of Failure Tasks / Total No of Tasks; Figure 5.5 depicts the rejection ratio of the project sorts with the project matter. When the project matter reaches 20 103, the rejection ratio of the project becomes saturated, and project kind five with a project matter of 20 103 has a small rejection ratio. The results show that as the project matter increases, the project rejection ratio gradually increases. It is possible to see that the proposed FTSMFFA has a very low rejection ratio. DFTST, FTSGA, and FTSACO have rejection ratios that are 2 to a few times higher than FTSMFFA. This is because it employs a fault-tolerant mechanism and a classification-based totally technique to map the most appropriate VM and host.

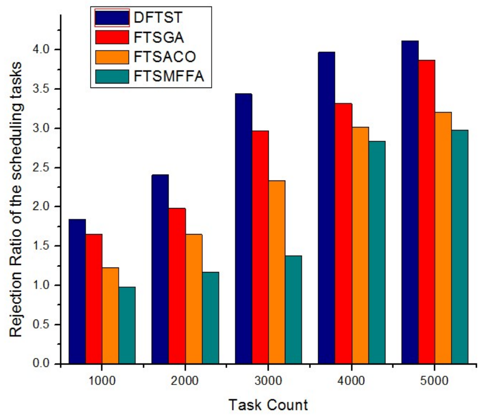


Figure 5.5: Rejection Ratio With Task Count

Energy Consumption: Energy consumption is the total amount of energy required to transmit the data packet from the one node to the other node.

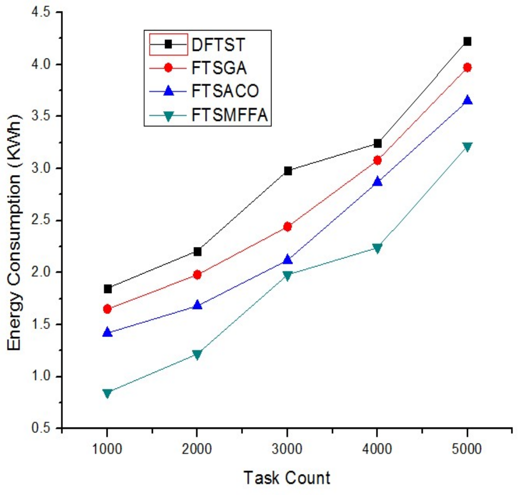


Figure 5.6: Energy Consumption With Task Count

here, this same daily calorie of something like the proposed work estimated and through direction after all technique of this same aspect of a varied struggle scheduling methods among these is already in comparing the with taking hold approaches is evidenced such as figure four.5.5. it and differentiate means that the facility dosage of something like the suggested FTSMFFA considerably lower since DFTST, FTSGA but rather FTSACO. Usually, that whole test uses up increased power than that of the flip side mainly are attributed types of obligations thanks to the fact that whole reminiscence demand but also cpu demand task is larger.

6. CONCLUSION

The study begins with a method for modeling cloud faults using fault tolerant scheduling algorithms. To represent the first firefly population, a variety of solutions were chosen. The fitness characteristic of the time table is considered when determining the execution time of each schedule. The schedules are passed in each iteration entirely based on the elegance of the schedule in order to create a new populace with better health. This study selects the time table with the best fitness in each iteration to find the best answers. The proposed FTSMFFA has shortened the time required to complete the submitted responsibilities. The simulation results using the FTSMFFA set of rules are to reduce the processing of typical execution with a cut-off date constraint, which is entirely based on the PB method of responsibilities. In the future, we can improve our fault tolerant scheduling strategy by taking into account risky computing environments. Another direction for our future work as cloud providers is to develop an energy efficient fault tolerant mission scheduling set of rules or different meta heuristic algorithms such as pso (particle swarm optimization), fish fly, artificial bee colony (abc), and cuckoo search.

REFERENCES

[1] . D. Breitgand and A. Epstein, “SLA-aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds”, IFIP/IEEE International Journal of Symposium on Integrated Network Management, Dublin, Ireland, vol 1, 161-168, 2011.

[2] . G. Mehta, E. Deelman, J. A. Knowles, T. Chen, Y. Wang, J. V¨ockler, S. Buyske, T. Matise, “Enabling data and compute intensive workflows in bioinformatics,” Journal of Euro-Par: Parallel Processing Workshops, 23-32, 2011.

[3] . X. Qin and H. Jiang, “A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems,” Journal of Parallel Computing sysytems, ISSN-978-1-4244-7386-1 vol-32, Issue-5, 331–356, 2006.

[4] . S. Ghosh, R. Melhem, and D. Moss_e, “Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems,” Journal of IEEE Trans. Parallel and Distributed Systems, vol 8, 3, 272–284,1997.

[5] . Zhang P, Zhou M (2017) “Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy” Journal of IEEE Transactions on Automation Science and Engineering ,15,2, 1–12, 2018.

[6] . J Wang , W Bao , X Zhu , et al. , FESTAL: fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized Clouds”,Journal of IEEE Transactions Computers. Vol 64,9,2015.

[7] . 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, 2013.

[8] . 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” Journal of IEEE Transaction Computers , vol 58,no 3, 380–393 , 2009.

[9] . Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S.”A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing.”Journal of Procedia Technology ,vol 10, 2013, 340-347.

[10] . Gu, J., Hu, J., Zhao, T., & Sun, G. (2012). “A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment.” Journal of Computers vol 7.no 1, 6563-6569. 2013.

[11] . GE Junwei., & YUAN Yongsheng, “Research of cloud computing task scheduling algorithm based on improved genetic algorithm.” Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), 347, ,2426-2429, 2013.

[12] . 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, no 4, 39-48. 2017.

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

[14] . Hajara Idris, 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”,Journal of Plos one , 2017.

[15] . X.-S. Yang, “Firefly algorithms for multimodal optimization”, Journal of Stochastic Algorithms: Foundations and Applications , 169-178, 2009.

[16] . A. Yousif, S. M. Nor, A. H. Abdullah, and M. B. Bashir, "A Discrete Firefly Algorithm for Scheduling Jobs on Computational Grid,” Journal of in Cuckoo Search and Firefly Algorithm, 271-290, 2014.

[17] . Zhu, X., Wang, J., Guo, H., Zhu, D., Yang, L. T., Liu, L, “Fault-Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds.” Journal of IEEE Transactions on Parallel and Distributed Systems, vol 27 ,no 12, 3501-3517, 2016.

[18] . Ali Nadhirah, Othman Mohd Azlishah ,” A Review of Firefly Algorithm” ARPN Journal of Engineering and Applied Sciences 2014 Asian Research Publishing Network (ARPN), vol 9,no 10, 2014.

[19] . Fanian, F., Bardsiri, V.K., & Shokouhifar, M. (2018). “A New Task Scheduling Algorithm using Firefly and Simulated Annealing Algorithms in Cloud Computing” Journal of Advanced Computer Science and Applications vol 3,no 9, 195-201, 2018.

[20] . X.S. Yang, "Firefly algorithm, stochastic test functions and design optimization," International Journal of Bio Inspired Computation vol 2,no 2, 78–84, 2010.