ISSN:2348-2079

Volume-5 Issue-1

International Journal of Intellectual Advancements and Research in Engineering Computations

Energy efficient scheduling and reduction of failure virtual machines with deadline constraints

Ms.V.Sivaranjani[1],S.Varshini[2],P.Vignesh[3],S.Subhasri[4]

Assistant Professor, UG Student, UG Student,UG Student

Department of computer scienceand Engineering

Kongu Engineering College, Perundurai.

varshusubramani@gmail.com

***Abstract*---** **Cloud computing is a scale-based computing model, and requires more physical machines and consumes an extremely large amount of electricity, which will reduce the profit of the service providers and harm the environment. It is shown that the cost of energy consumed by a server during its lifetime will exceed the cost of server itself. Virtual machine scheduling is one of the most important and efficient technologies of reducing energy consumption in cloud. The main idea of scheduling VMs energy efficiently is placing them on only part of the physical machines and transforming the other ones into low power state (sleep or off). Existing energy efficient scheduling methods of virtual machines (VMs) in cloud cannot work well if the physical machines (PMs) are heterogeneous and their total power is considered, and typically do not use the energy saving technologies of hardware, such as dynamic voltage and frequency scaling (DVFS).Also, datacenters have to reduce the number of failed virtual machines (tasks) in order to achieve high quality service. To avoid these kinds of issues, this paper proposes an algorithm which reduces energy consumption in data centers and extends the deadline of virtual machines. The algorithm works as follows: There exists optimal frequency for a PM to process certain VMs, based on which the notion of optimal performance–power ratio is defined to weight the homogeneous PMs. The PM with higher optimal performance–power ratio will be assigned to VMs first to save energy. The process is divided into some equivalent schedule periods, in each of which VMs are allocated to proper PMs and each active core operates on the optimal frequency. After each period, the cloud should be reconfigured to consolidate the computation resources to further reduce the energy consumption and check for enough resources to complete the tasks within deadline. VMs having tasks which are not completed within the deadline can ask the user to extend th deadline.If yes, reconfigure the cloud and allocate the incomplete VMs to the appropriate PM. Our proposed work reduces energy consumption, number of failed virtua machines effectively.**

***Keywords*---** **VM scheduling, Energy efficiency, Cloud computing, Deadline constraints.**

I. INTRODUCTION

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. As the data centers increases, consumption of energy in data centers has become a great Virtualization is an important technology typically adopted in cloud to consolidate the resources and support the pay-as-you-go service paradigm. It has been reported that virtual machines could be used for scientific applications with tolerable performance punishment, and could provide desirable, on demand computing environments for any users. Virtual machine scheduling is one of the most important and efficient technologies of reducing energy consumption in cloud. This paper focuses on dynamic scheduling of virtual machines to achieve energy efficiency and satisfy deadline constraints in the cloud with heterogeneous physical machines. Then the notion of optimal performance–power ratio is defined to weight the heterogeneities of the physical machines, VMs will be allocated prior to the PMs with higher optimal performance–power ratio. The scheduling is divided into some equivalent periods, and the cloud will be reconfigured after each period to consolidate the computation resource to further reduce the energy consumption. Finally, the deadline constraint is maintained by the definition of required resource, VM can be completed on time as long as it is allocated successfully to a PM.

II. Related works

Virtualization is widely used in cloud computing to fully utilize the resources and improve the performance. Various VM scheduling methods [1-3] have been proposed to dynamically allocate and consolidate the VMs in cloud computing environment. The allocation algorithms can be mainly divided into two types, allocating VMs onto PMs and assigning PMs to VMs. The consolidation is typically achieved by VM migrations. Energy consumption was not considered in traditional VM scheduling in cloud computing. Energy efficient VMs scheduling in data centres mainly focuses. There are two main challenges for energy efficient VM scheduling in cloud environment, heterogeneous PMs and practical energy consumption of the PMs. The assumption of homogeneous PMs is often adopted in energy efficient VM scheduling in cloud computing, which is not actual in practice. The energy of processors is often used to replace the energy of PMs for scheduling of computation intensive VMs, while low energy of processors does not mean low energy of physical machines especially for heterogeneous PMs. One of the first studies of energy efficient VMs scheduling using DVFS in clusters was proposed in [4]. Only the energy consumed by the processors is measured, which is not enough to reflect the energy efficiency of a cluster, because processors only consume about a quarter of total energy consumption, and the ratio will be even smaller when the operating frequency is low. Energy consumption in data centers have been greatly reduced by finding optimal performance power ratio for all physical machines and then allocating virtual machines prior to machines having high optimal performance power ratio. But the disadvantage in this paper is the failure of virtual machines is comparatively high due to the deadline constraints. Our paper greatly reduces the failure of virtual machines by extending the deadline and reconfiguring the cloud.

III. System model

3.1. Power model

The power consumption of modern processors can be divided into two parts, dynamic power and static power, $P_{cpu} = P_{static} + P_{dynamic}$.The dynamic power $P_{dynamic} = aCV^2f$, where a is the switching activity, C is the physical capacitance, V is the supply voltage and f is the operating frequency. The values of switching activity and capacitance can be viewed as constants because they are only determined by the low-level system design The total power of a PM is mainly consumed by

the processor, memory, disk and other components. When processing the computation-intensive tasks, the power consumption of a PM is assumed to be $P = P_{cpu} + P_s$ in this paper, where P_{cpu} and P_s are the power of processor and all other components including memory, disk and so on. Additionally, we assume that P_s is fixed when the VMs are computation-intensive. The energy of the whole physical machine is equation

$$E = (P_s + P_{static} + P^{f(j)}_{dynamic})t.$$

where t is the execution time of the core of processor *cpu*, and t=0 if the core is idle. Obviously, the energy caused by P_s and P_{static} is determined by the maximum execution time of the cores, since the processor and other components should be activated as long as any core is active.The static power is mainly determined by the type of transistors and the process technology. It is reported that the idle power of a processor may sometimes exceed 50% of the peak power, and the main part of the idle power is the static power. Therefore, we suppose that $P_{static} = \beta P^{max}_{dynamic}$, $(0 < \beta \leq 1)$. For the simplicity of descriptions, we assume that $Ps = \gamma P^{max}_{dynamic}$, $(\gamma > 0)$. Then we can get the total power and energy of a PM, $P = P_{dynamic} + P_{static} + P_s$, $E = (P_{dynamic} + P_{static} + P_s) t$, where t is the execution time for the PM to process the computation intensive VMs. The execution time is affected by the computation of the VM and the operating frequency of the core, i.e. t = w/f, where w is the computation of the VM.

3.2. Physical machines model

The set of heterogeneous PMs is $PM = \{pm_1, pm_2, \ldots, pm_n\}$, where n is the number of PMs. Each PM is divided into two parts, the processor and other components, and the power of the former P_{cpu} varies on different frequencies, while that of the latter P_s keeps stable. The processor of each PM is defined as $cpu = (nc, ns, P_{static}, F, P)$, where nc and ns are the number of cores and the number of operating states of the processor, P_{static} is the static power of cpu, F and P are the sets of operating frequencies and dynamic power respectively. We assume that the nc cores of PM pm are homogeneous, so $F=\{f_1,f_2,\ldots,f_{ns}\}$,$P=\{p_1,p_2,\ldots,p_{ns}\}$F={f1,f2,…,fns},P

={p1,p2,…,pns}, where fj and pj are the frequency and dynamic power of processor cpu_i jth operating status. Without loss of generality, we suppose $f_1<f_2<\cdots<f_{ns}$f1<f2<⋯<fns and $p_1<p_2<\cdots<p_{ns}$p1<p2<⋯<pns, and the minimal and maximal of the operating frequency and dynamic power are f_{min}=f1,f_{max}=fns and $p_{\min}=p_1, p_{\max}=p_{ns}$. And if the core sleeps, its frequency and dynamic power are set to zero. Therefore, the total energy consumption of the cloud for processing given tasks is equation(2)

$$E=\sum_{i=1}^{n} y(i)\left(\left(P_{i,s}+P_{i,static}\right)\max_{j=1}^{nc_i}\{t_{i,j}\}+\frac{1}{nc_i}\sum_{j=1}^{nc_i} x(i,j)\,P_{i,dynamic}^{f(i,j)}t_{i,j}\right)$$

The static power, also called leakage power, is caused by leakage currents which are present in any active circuits.

3.3. Virtual machine model

We focus on computation-intensive VMs which arrive continuously to be processed on the cloud in this paper. The set of VMs is $VM=\{vm_1,vm_2,\ldots,vm_m\}$VM={vm1,vm2,…,vmm} and each virtual machine vmi is assumed to be processed by a core at any time, i.e. vm_i cannot run on two or more cores at the same time. The execution time and power consumption of migrations of the virtual machine are ignored in this paper.

IV.Energy efficient scheduling of virtual machines

4.1. Finding optimal performance power ratio

Optimal performance–power ratio, oppr, for a physical machine pm is the ratio of the computation resource to the power of pm when all cores are operating on optimal frequency. It can be computed as pm.oppr=pm.nc*f_{opt} /pm.P_s+pm.P_{static}+pm.$P_{dynamic}$ ^{fopt} where fopt is computed by using $\phi* = h\sqrt{(\beta+\gamma)/(h-1)}$, that is all cores of pm are active.

4.2.Existing Scheduling algorithm

Algorithm 1. EEVS

Input: set of physical machines PM, set of virtual machines VM
Output: schedule of VM, energy consumption, processing
time

```
1 sort PM with decreasing oppr;
2 foreach period t do
// VMAllocation, shown in Algorithm 2
3 allocate each VM in VM(t);
// VMProcess, shown in Algorithm 3
4 set frequency of each active PM;
5 update the information of active PMs and running VMs;
// Reconfiguration, shown in Algorithm 4
6 reconfigure the cloud;
7 if all VMs are finished or failed then
8 return Schedule, Etotal, Ttotal;
```

Algorithm 2. VMAllocation

```
1 foreach vm in VM(t) do
2 foreach pm in PM do
3 if pm has idle cores then
4 an idle core is allocate for vm;
5 else
6 foreach core of pm do
7 if core has enough resource for vm then
8 core is assigned to vm;
9 AllocationList.add(vm, pm, core);
10 vm.st = t;
11 core.w+ = vm.rr;
12 if vm is not allocated successfully then
13 update vm.rr;
14 insert vm into the head of VM(t + 1);
15 return AllocationList;
```

Algorithm 3. VMProcess

```
1 foreach pm in PM do
2 foreach core in pm do
3 if core is active then
4 SetOptFrequency(core);
5 foreach (vm, pm, core) in AllocationList do
6 if vm is completed in this period then
7 core.w− = vm.rr;
8 pm.w− = rm.rr;
9 if core.w = 0 then
10 translate core into sleep state;
11 if pm.w = 0 then
12 translate pm into sleep state;
13 AllocationList.delete(vm, pm, core);
14 return AllocationList;
SetOptFrequency(pm, core)
15 foreach pm in PM do
16 count active cores of pm;
17 compute fopt for pm using Eqs. (4)–(6);
18 foreach active core in pm do
19 if core.w ≥ fopt then
20 core.f = min{fi|fi > core.w & fi ∈ pm.F };
21 else core.f = fopt ;
22 return core.f ;
```

Algorithm 4. Reconfiguration

```
1 foreach pm in PMIoppr do
2 foreach core in pm do
```

```
3 for each vm on core do
4 (pm′, core′) = MigrateVM(vm, pm);
5 if pm ≠ pm′ then
6 update: core.w− = vm.rr, pm.w− = vm.rr;
7 update: core′.w+ = vm.rr, pm′.w+ = vm.rr;
8 AllocationList.delete(vm, pm, core);
9 AllocationList.add(vm, pm′, core′);
10 return AllocationList;
MigrateVM(vm, pm)
11 foreach pm′ with ppr higher than pm in PM do
12 foreach core′ in pm do
13 if core′ has enough resource for vm then
14.return(pm’,core’)
15.return(pm,vm)
```

Proposed algorithm

Algorithm 1. EEVS-FVM

```
Input: set of physical machines PM, set of virtual machines VM
Output: schedule of VM, energy consumption, processing time
1 sort PM with decreasing oppr;
2 foreach period t do
// VMAllocation, shown in Algorithm 2
3 allocate each VM in VM(t);
// VMProcess, shown in Algorithm 3
4 set frequency of each active PM;
5 update the information of active PMs and running VMs;
// Reconfiguration, shown in Algorithm 4
6 reconfigure the cloud;
7 if all VMs are finished or failed then
8 return Schedule, Etotal, Ttotal;
```

Algorithm 2. VMAllocation

```
1 foreach vm in VM(t) do
2 foreach pm in PM do
3 if pm has idle cores then
4 an idle core is allocate for vm;
5 else
6 foreach core of pm do
7 if core has enough resource for vm then
8 core is assigned to vm;
9 AllocationList.add(vm, pm, core);
10 vm.st = t;
11 core.w+ = vm.rr;
12 if vm is not allocated successfully then
13 update vm.rr;
14 insert vm into the head of VM(t + 1);
15 return AllocationList;
```

Algorithm 3. VMProcess

```
1 foreach pm in PM do
2 foreach core in pm do
3 if core is active then
4 SetOptFrequency(core);
5 foreach (vm, pm, core) in AllocationList do
6 if vm is completed in this period then
7 core.w− = vm.rr;
8 pm.w− = rm.rr;
9 if core.w = 0 then
10 translate core into sleep state;
11 if pm.w = 0 then
12 translate pm into sleep state;
13 AllocationList.delete(vm, pm, core);
14else
15Extend the deadline of tasks
16do steps in algorithm4
17 return AllocationList;
SetOptFrequency(pm, core)
18 foreach pm in PM do
19 count active cores of pm;
20 compute fopt for pm using Eqs. (4)–(6);
21 foreach active core in pm do
22 if core.w ≥ fopt then
23 core.f = min{fi|fi > core.w & fi ∈ pm.F };
24 else core.f = fopt ;
25 return core.f ;
```

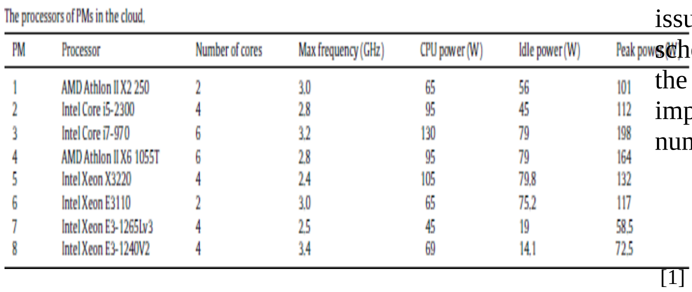
Algorithm 4. Reconfiguration

```
1 foreach pm in PMIoppr do
2 foreach core in pm do
3 for each vm on core do
4 (pm′, core′) = MigrateVM(vm, pm);
5 if pm = pm′ then
6 update: core.w− = vm.rr, pm.w− = vm.rr;
7 update: core′.w+ = vm.rr, pm′.w+ = vm.rr;
8 AllocationList.delete(vm, pm, core);
9 AllocationList.add(vm, pm′, core′);
10 return AllocationList;
MigrateVM(vm, pm)
11 foreach pm′ with ppr higher than pm in PM do
12 foreach core′ in pm do
13 if core′ has enough resource for vm then
14.return(pm’,core’)
15.return(pm,vm)
```

IV. Experimental setup

We use cloudsim for the simulation of this algorithms. Cloud computing is the infrastructure, including the data centers, networking, and communication standards.Cloud operations are the management tools, the APIs, and the many disciplines associated with managing the cloud environments.

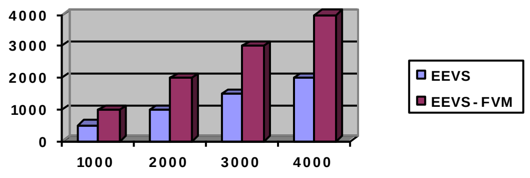
PHYSICAL MACHINE SPECIFICATION:



The processors of PMs in the cloud.

PM	Processor	Number of cores	Max frequency (GHz)	CPU power (W)	Idle power (W)	Peak pow (W)
1	AMD Athlon II X2 250	2	3.0	65	56	101
2	Intel Core i5-2300	4	2.8	95	45	112
3	Intel Core i7-970	6	3.2	130	79	198
4	AMD Athlon II X6 1055T	6	2.8	95	79	164
5	Intel Xeon X3220	4	2.4	105	79.8	132
6	Intel Xeon E3110	2	3.0	65	75.2	117
7	Intel Xeon E3-1265Lv3	4	2.5	45	19	58.5
8	Intel Xeon E3-1240V2	4	3.4	69	14.1	72.5

V.ExperimentalResults



VI. Conclusion

We can find that the main challenges for energy efficient scheduling of VMs in cloud computing are the heterogeneity of the PMs, the total power consumption of each PM. To overcome these issues, we propose algorithm, an energy efficient scheduling algorithm of virtual machines to reduce the total energy consumed by the cloud by implementing this algorithm ,we can reduce the number of failed VMs as high as possible.

VII. REFERENCES

[1] G. Laszewski, L. Wang, A.J. Younge, X. He, Power-aware scheduling of virtual machines in DVFS-enabled clusters, in: IEEE International Conference on Cluster Computing and Workshops, 2009, pp. 1–10.

[2] Zhen Xiao, Weijia Song, Qi Chen, Dynamic resource allocation using virtual machines for cloud computing environment, IEEE Trans. Parallel Distrib. Syst.24 (6) (2013) 1107–1117.

[3] S. Bazarbayev, M. Hiltunen, K. Josh, Content-based scheduling of virtual machines (VMs) in the cloud, in: International Conference on Distributed Computing Systems, 2013, pp. 93–101.

[4] I. Hwang, M. Pedram, Hierarchical virtual machine consolidation in a cloud computing system, in: IEEE Sixth International Conference on Cloud Computing, 2013, pp. 196–203.