Cloud Computing CPU Allocation and Scheduling Algorithms Using CloudSim Simulator

FCFS

- Algorithm is based on the arrival time of the resource request.
- AwT = (∑ Tn) / NT
- Where AwT - Average Waiting Time, Tn: Tasks Waiting time for execution, NT: Number of tasks

SJF

- Choose the task with the shortest execution time in order to take the lead of the queue and then FCFS way.
- AwT = (∑ Tn) / NT
- Where AwT - Average Waiting Time, Tn: Tasks Waiting time for execution, NT: Number of tasks

RR (static time quantum)

- Tasks submitted to VM sorted in ascending order based on burst time (execution time for each task)
- Computing time quantum
  - TQ = (NP * MIPS) / 1000
    - TQ (Time Quantum), NP (Number of Processors), MIPS (Millions Instruction per second)
- For each task in the queue list
  - CPU allocates the time quantum for task execution
  - If task executed, sent to finished list
  - If task not executed, sent to waiting list.

RR (dynamic time quantum)

- Tasks submitted to VM sorted in ascending order based on burst time (execution time for each task).
- Computing time quantum
  - TQ = ∑ Tn / NT
    - TQ (Time Quantum), Tn (task burst time), NT(Number of tasks)
- For each task on the queue list
  - If task burst time < time quantum
    - Time quantum = task burst time
    - CPU allocates time quantum for task execution
    - Task executed, sent to finish list
  - If task burst time > time quantum
    - CPU allocates time quantum for task execution
    - Task sent to the waiting list
  - If waiting list not empty
    - Send tasks from waiting list to queue list.

Analysis of Job Scheduling Algorithms in Cloud Computing

FCFS

- Initialize tasks
- First task assigned to the queue and add task up to n numbers
- Add next tan in the main queue

SJF

- for i = 0 to i < main queue-size
    - if task **i+1** length < task **i** length
        - add task i+1 in front of task i in the queue

Improved Max-Min Algorithm in Cloud Computing

Max-Min

1. for all submitted tasks in meta-task; T_i
2. for all resources; R_j
3. Cij = Eij + rj
4. While meta-task is not empty
5. find task $T_k$consumes *maximum completion time*.
6. assignT_k to the resource R_j which gives *minimum execution time*.
7. remove $T_k$from meta-tasks set
8. update rj for selected Rj
9. update Cij for all j

Improved Max-Min

1. for all submitted tasks in meta-task; T_i
2. for all resources; R_j
3. Cij = Eij + rj
4. While meta-task is not empty
5. find task $T_k$costs *maximum execution time*.
6. assignT_k to the resource R_j which gives *minimum completiontime.*
7. remove T_k from meta-tasks set
8. update $r_j$for selected R_j
9. update Cij for all j

COMPARATIVE STUDY OF SCHEDULING AL-GORITHMS IN CLOUD COMPUTING ENVIRONMENT

Random Algorithm

```
Nocl ← cloudletlist.size();
NoVM ← VML.size();
index ← 0;
for j ← 0 to Nocl do
cl ← cloudletlist.get(j);
index ← random() × (NoVM − 1);
v ← VML.get(index);
stagein ← TransferTime(cl, v, in);
stageout ← TransferTime(cl, v, out);
exec ← ExecuteTime(cl, v);
if (cl.AT + stagein + exec + stageout + v.RT ≤ cl.DL) then
sendjob(cl, v);
update(v);
else
Drop(cl);
FailedJobs;
endif
```

Round Robin algorithm

```
Nocl ← cloudletlist.size();
NoVM ← VML.size();
index ← 0;
for j ← 0 to Nocl do
cl ← cloudletlist.get(j);
index ← (index+1) mod NoVM;
v ← VML.get(index);
stagein ← TransferTime(cl, v, in);
stageout ← TransferTime(cl, v, out);
exec ← ExecuteTime(cl, v);
if (cl.AT + stagein + exec + stageout + v.RT ≤ cl.DL) then
sendjob(cl, v);
update(v);
else
Drop(cl);
FailedJobs;
end
```

Minimum Completion Time

```
initialization;
Nocl ← cloudletlist.size();
NoVM ← VML.size();
index ← 0;
for j ← 0 to Nocl do
cl ← cloudletlist.get(j);
min ← +∞;
for i ← 0 to NoVM do
v ← VML.get(i);
if min > (v.getready() + cl.getlength()/v.speed) then
min ← v.getready() + cl.getlength()/v.speed;
index ← i;
end
end
v ← VML.get(index);
stagein ← TransferTime(cl, v, in);
stageout ← TransferTime(cl, v, out);
exec ← ExecuteTime(cl, v);
if (cl.AT + stagein + exec + stageout + v.RT ≤ cl.DL) then
sendjob(cl, v);
update(v);
else
Drop(cl);
FailedJobs;
end
```

Opportunistic Load balancing algorithm

```
initialization;
Nocl ← cloudletlist.size();
NoVM ← VML.size();
index ← 0;
for j ← 0 to Nocl do
cl ← cloudletlist.get(j);
min ← +∞;
for i ← 0 to NoVM do
v ← VML.get(i);
if min > (v.getready()) then
min ← v.getready();
index ← i;
end
end
v ← VML.get(index);
stagein ← TransferTime(cl, v, in);
stageout ← TransferTime(cl, v, out);
exec ← ExecuteTime(c, v);
if (cl.AT + stagein + exec + stageout + v.RT ≤ cl.DL) then
sendjob(cl, v);
update(v);
else
Drop(cl);
FailedJobs;
end
```

RASA: A New Task Scheduling Algorithm in Grid Environment

<u>Min-Min</u>

1. **for** all tasks T_i in meta-task M_v
2. **for** all resources R_j
3. $C_{ij}=E_{ij}+r_j$
4. **do** until all tasks in M_v are mapped
5. for each task in M_v find the earliest completion time and the resource that obtines it
6. find the task T_k with the ***minimum*** earliest completion time
7. assigne task T_k to the resource R_l that gives the earliest completion time
8. delete task T_k from M_v
9. update r_l
10. update C_{il} for all i
11. **end do**

<u>Max-Min</u>

1. **for** all tasks T_i in meta-task M_v
2. **for** all resources R_j
3. $C_{ij}=E_{ij}+r_j$
4. **do** until all tasks in M_v are mapped
5. for each task in M_v find the earliest completion time and the resource that obtines it
6. find the task T_k with the maximu earliest completion time
7. assigne task T_k to the resource R_l that gives the earliest completion time
8. delete task T_k from M_v
9. update r_l
10. update C_{il} for all i
11. **end do**

Resource aware scheduling algorithm

```
for all tasks T_i in meta-task M_v
    for all resources R_j
        C_ij=E_j+r_j
do until all tasks in M_v are mapped
    if the number of resources is even then
        for each task in M_v find the earliest completion
            time and the resource that obtines it
        find the task T_k with the maximum earliest
            completion time
        assigne task T_k to the resource R_l that gives the
            earliest completion time
        delete task T_k from M_v
        update r_l
        update C_il for all i
    else
        for each task in M_v find the earliest completion
            time and the resource that obtines it
        find the task T_k with the minimum earliest
            completion time
        assigne task T_k to the resource R_l that gives the
            earliest completion time
        delete task T_k from M_v
        update r_l
        update C_il for all i
    end if
end do
```

Efficient Task Scheduling Algorithms for Cloud Computing Environment

Longest Cloudlet Fastest Processing element (LCFP)

- Sort the cloudlets in descending order of length.
- Sort the PEs across all the hosts in descending order of processing power.
- Create virtual machines in the sorted list of PEs by packing as many VMs as possible in the fastest PE.
- Map the cloudlets from the sorted list to the created VM.

Shortest Cloudlet Fastest Processing element (SCFP)

- Sort the cloudlets in ascending order of length.
- Sort the PEs across all the hosts in descending order of processing power.
- Create virtual machines in the sorted list of PEs by packing as many VMs as possible in the fastest PE.
- Map the cloudlets from the sorted list to the created VM.

Task Scheduling in Cloud Computing: A Survey

A Comparison Study of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Ileterogeneous Distributed Computing Systems

A SURVEY ON ECONOMIC CLOUD SCHEDULERS FOR OPTIMIZED TASK SCHEDULING

Study and Analysis of Various Task Scheduling Algorithms in the Cloud Computing Environment

Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems