Abstract

Objectives: The complex problems are solved by sharing the resources in the grid environment. The proposed ORR approach ensures the availability of resources when it is required in the grid computing environment. The ORR approach also strives to reduce the process switching which is high in TARR (Time-Slice based Resource Reservation). Methodology: The ORR maintains slice queue in which the free time slots are entered. If the free time slices are taken as such then there increase in overhead on process switching. So, the selectslicequeue maintains the slices in best-fit approach based on the current request. In this paper, samples of ten jobs in two scenarios are considered. It is compared with the existing FCFS (First Come First Serve) basis and TARR approach. Findings: In the existing FCFS approach, there is resource reservation denial because of non-availability of resources at the time slot requested. Though the resources are reserved in time-slice basis in TARR based on the defer time, the number of process switching is high. In this case even the small chunk of time-slice is used. The increased process switching increases the overhead on process suspension and resumption. Hence the introduction of best-fit strategy in ORR approach reduces the reservation denial of FCFS and also reduces the overhead on process suspension and process resumption as in TARR. Applications: The ORR scheme can be incorporated in the existing grid environment as Globus, Nimrod G, Legion etc., New agreed virtual organizations can also make use of the ORR scheme.

Keywords: Context Switching, Grid Computing, Optimal Resource Reservation, Resource Management, Slice Queue, TLB

1. Introduction

Grid computing is defined as sharing of resources among the agreed virtual organizations to enhance the performance of the processes in the environment. In grid, the resources are geographically distributed in multiple administrative domains, managed and owned by different organizations with different policies and interconnected by wide area networks or the internet. Based on the tasks, the grid can be classified as computational grid, data grid, service grid etc. Grid Computing can also be used to integrate the data in educational institutions which is the data grid. Irrespective of the type of grid the resource management plays a vital role in deciding the performance of the grid environment. There were various methods for resource management and in an approach is proposed where the resources are managed by grouping on QoS and Semantics.

To accomplish the assigned task more efficiently, the resources are reserved in advance, which guarantees the availability of resource when required. There were various advance resource reservation schemes available as FCFS, priority based reservation, Alternate Offer Protocol and TARR.

TARR provides a better approach by reserving the resources even if the entire duration is not available.
The interleaving time slices are considered and they were reserved in TARR. But this approach suffers drawbacks as there is no restriction made in the number of time slices. If small chunks of time are allocated then the processing overhead for suspension and resuming would increase.

The optimal strategy is used in selecting the free slots. In section 2, the proposed ORR approach is explained, which comprises of the algorithms and section 3 deals with the performance metrics. Section 4 deals with the comparative analysis, which is followed by the conclusion.

ARR in grid computing environment provides a guaranteed availability of the resources for the task to be completed. Various researches are going on in this advance resource reservation. All the reservation schemes take the Start Time (ST) and Finish Time (FT) of the resource as the input to make the reservation.

In the negotiation mechanism between the proposer and the responder is proposed. The resource requester is the proposer and the resource holder is the responder. The resource requester sends the request and a unique negotiation id is agreed between the proposer and responder. Then the proposal is submitted which can be accepted or rejected. All these operations are performed by the proposer and the responder, there are no time-limits or any constraints imposed in the protocol.

In the reservation based on the priority is assigned. In this approach the advance reservations are made on priority basics whenever tasks with same start time request for the resource. Always the reservation request comes with the priority. And the system tries to make the at most profit by providing the resources to the high priority request.

The resource reservation for an opportunistic computing environment is proposed. When a resource reservation is made then the available nodes are checked for the free slots. If free slots are available then the reservation is done. If free slots are not available during the current reservation request then the next available free slot is reserved.

In the advance resource reservation based on the FCFS approach is proposed. When a resource is requested for reservation then the reservation can be done on the First Come First Serve (FCFS) basis. A list is maintained at each resource for reservation. When a new request for reservation comes in then the list is checked with the start time and finish time of the existing reservations and new request. If it is possible to fit in the reservation within the empty slots then the reservation can be done otherwise the reservation is rejected.

In the TARR (Time Slice based Advance Resource Reservation) which reserves the resource when time slice is free, by including defer time (DT) which is the time until the job can be completed is proposed.

In all the existing approaches only when the resource is available in the specified start time and finish time the reservation is done. Because of this even if the resource is available for short duration than expected then the resource is kept unused. The ORR approach tries to remove this drawback by allowing the usage of time-slices as in TARR approach and by applying the best fetch strategy.

2. Optimal Resource Reservation.

In the proposed ORR (Optimal Resource Reservation) approach the best fetch strategy is considered. During reservation, if the slots requested are empty then they are reserved. The conflict occurs only when the slots are not available. Normally, the reservation denial is done in FCFS approach In TARR, the free slots are considered and the reservation slots are provided rather than as a single slot. For this purpose, in addition to the proposed start time and finish time, the defer time (DT) is also considered. The defer time is time until which the job can be completed or considered for reservation. In TARR approach, whenever a free slot is available it is allotted as such. TARR necessitates more context switching i.e., whenever a small chunk of time slice is available, then that is reserved which requires more process suspension and resumption. The sequence diagram in Figure 1 depicts the context switching.
The entity resource can be reserved for a period of time. After the elapse of time the current process in execution need to undergo process switching. The current status of the PCB (Process Control Block) is stored. And the detail on process to be resumed is retrieved. The new process possesses the resource. The state transition or process switching causes additional overhead. Hence in this proposed ORR the slicequeue and selectslicequeue are maintained.

2.1 Slice Queue

The reservation list of a resource maintains the details on resource and also the job which has reserved the respective resource. Figure 2 depicts the resource list with reservation and empty slots.

To find the empty slice initially the reservation list is taken as input. The reservation list consists of the job id which has reserved the resource with its period of reservation i.e., its start time and finish time. The free slots are identified. The difference between the finish time of the previous reservation and the start time of the current reservation is computed. If there exist any time then that free slice is inserted into the slice queue.

Thus the slice queue maintains the free time slices. The Slice queue consists of slice id, the slice start time (SST) and the slice finish time (SFT) of a free slice. The following algorithm inserts the free time slices into the queue.

Algorithm SliceQueue_Insert(reserve_list)
Begin
front = 0, rear = 0
for i = 0 to list_size - 1
   timeslice = start\_i+1 - finish\_i
   if timeslice > 0 then
      sst = finish\_i
      sft = start\_i+1
      slicequeue[rear] = {sid, SST, SFT}
   end if
end for
while considering the Figure 2, the free slots from 3 to 5 and from 10 to 16 are placed in the slice queue. The Slice queue maintains all the free time slots. The time slots are taken in first free slot available approach as in TARR°. The slice queue for Figure 2 is shown in Figure 3. This leads to more context switching which leads to increase in overhead.

2.2 Select Slice Queue

After generating the slice queue, the slice is searched for reservation. The slice queue consists of all the free slices. But not all the free slices can be used. Hence, the select slice queue algorithm finds the slice which is more relevant to the required slot. The algorithm considers the free slot and the Defer Time (DT) to fetch the slots.

Algorithm selectslicequeue(slicequeue)
begin
if slicequeue is empty then
   return no_slice
else
   for i = 0 to size(slicequeue)
      if (slicequeue.sft < DT) then
         insert selectslicequeue(sid, slicequeue.sst, slicequeue.sft)
      end if
   end for
end if
end

The selectslicequeue algorithm selects the slots only when the time slice is within the defer time.

2.3 The ORR Algorithm

The ORR algorithm finds the optimal time slice from the selectslicequeue. For this the algorithm takes in the job id of the process and the selectslicequeue as input. And it returns the optimal reservation of resource which is explained in Figure 3.
The selectslicequeue maintains the time slices that can be reserved. The size function in ORR returns the number of elements in the particular queue.

Initially the time required for the job is estimated. It is computed by calculating the difference between the finish time of that job and the start time of the job. Then the found value is set to zero. It is used as a flag variable. If it is left as zero then it means that the reservation is not made possible. Only when it is turned to one the slot is available.

The size of the free slice is computed from the selectslicequeue. Since the selectslicequeue maintains the slice start time and slice finish time the difference between the two is computed and maintained in slot variable.

In this algorithm if the first slice itself is in required size then it is allocated and it is the optimized allotment as it doesn’t require the context switching. Otherwise the slot is allotted and the required time is reduced by the slot value. So that the required time value contains the required time slot after partial reservation.

The found value indicates the number of time slices considered for reservation for the particular job id. The found value is compared with the threshold value. If the found is greater than the threshold value then the randomize function is called on the selectslicequeue. This function shuffles the time slices and the time slices are selected. When the numbers of time slices are below the threshold value then the selected slices are assigned for the job.

Algorithm ORR(JID, SelectSliceQueue)
begin
    requiredtime = FT – ST
    found = 0
    for i = 1 to size(selectslicequeue)
        slot = selectslicequeue.sft - selectslicequeue.sst
        if (slot >= requiredtime && found = 0) then
            allot(sid, selectslicequeue.sst, selectslicequeue.sft)
            return
        else
            requiredtime = requiredtime – slot
            allot(sid,selectslicequeue.sst,selectslicequeue.sft)
            found = found + 1
        end if
    end for
    if found > T then
        randomize(selectslicequeue)
        ORR(JID, selectslicequeue)
    else
        assignslice(found)
    end if
end

3. Performance Metrics

The metric considered for evaluation of the algorithm is the Overhead on suspension and resuming of process (OH) which is measured by Number of Switches (NoS).

3.1 Overhead on Suspension and Resuming (OH)

If small time slices are used for reservation the overhead on process suspension and resumption would be high. To avoid this overhead the near optimal time slice is chosen for execution. For this the number of process switches is reduced. The hardware that supports the process switching is the TLB Translation Lookaside Buffer. Whenever the process switching is required the TLB is searched for the process information. When the process information is available then it is the TLB hit otherwise it is TLB miss. In case of TLB miss the page walk is performed to retrieve the information, hence this requires more overhead.

4. Comparative Analysis

To compare the algorithms as FCFS, TARR and the proposed approach the sample scenarios are considered. In Table 1 JID refers to the job – id, ST and FT refers to the expected start time and finish time of the respective job. The DT refers to the Defer Time until which the job can be postponed.

| Table 1. Scenario 1 & 2 |
|-------------------------|
|                         |
| **SCENARIO 1**          | **SCENARIO 2** |
| JID | ST | FT | DT | JID | ST | FT | DT |
| J1  | 3  | 7  | 20 | J1  | 1  | 4  | 10 |
| J2  | 9  | 12 | 20 | J2  | 4  | 6  | 12 |
| J3  | 6  | 9  | 25 | J3  | 8  | 10 | 20 |
| J4  | 15 | 19 | 25 | J4  | 5  | 8  | 16 |
| J5  | 22 | 24 | 30 | J5  | 12 | 15 | 25 |
| J6  | 27 | 30 | 35 | J6  | 18 | 19 | 29 |
| J7  | 32 | 33 | 40 | J7  | 20 | 23 | 33 |
| J8  | 20 | 23 | 30 | J8  | 21 | 25 | 35 |
| J9  | 35 | 40 | 45 | J9  | 25 | 28 | 38 |
| J10 | 42 | 44 | 50 | J10 | 29 | 31 | 41 |
In Scenario 1, when FCFS is followed the J1 and J2 jobs can be reserved and it is shown in Table 2. When J3 requests for reservation, it cannot be provided. Since J1 is reserved during that time. J3 is denied as a consequence. Similarly J8 is also denied in scenario 1, as J5 is reserved during that period.

Table 2. Specifying the FCFS based reservation

| SCENARIO 1 | SCENARIO 2 |
|------------|------------|
| Allot      | ST  | ET  | Allot  | ST  | ET  |
| J1         | 3   | 7   | J1     | 1   | 4   |
| J2         | 9   | 12  | J2     | 4   | 6   |
| J4         | 15  | 19  | J3     | 8   | 10  |
| J5         | 22  | 24  | J5     | 12  | 15  |
| J6         | 27  | 30  | J6     | 18  | 19  |
| J7         | 32  | 33  | J7     | 20  | 23  |
| J9         | 35  | 40  | J9     | 25  | 28  |
| J10        | 42  | 44  | J10    | 29  | 31  |

In TARR, there is no resource request denial as the time slices are considered for the reservation and it is shown in Table 3. Hence Job J3 is assigned from time slice 7 to 9 and 12 to 13. This requires the job to be suspended and resuming back. This requires more overhead. Similarly Job J8 is also allotted in TARR, which is denied in FCFS. J8 is assigned between 20 and 22 and after an elapse of time it is allotted from 24 to 25. Similarly in scenario 2 also the job J4 is assigned in two time slots and job J8 is assigned thrice.

Table 3. Specifying the TARR based reservation

| SCENARIO 1 | SCENARIO 2 |
|------------|------------|
| Allot      | ST  | ET  | Allot  | ST  | ET  |
| J1         | 3   | 7   | J1     | 1   | 4   |
| J3         | 7   | 9   | J2     | 4   | 6   |
| J2         | 9   | 12  | J4     | 6   | 8   |
| J3         | 12  | 13  | J3     | 8   | 10  |
| J4         | 15  | 19  | J4     | 10  | 11  |
| J8         | 20  | 22  | J5     | 12  | 15  |
| J5         | 22  | 24  | J6     | 18  | 19  |
| J8         | 24  | 25  | J7     | 20  | 23  |
| J6         | 27  | 30  | J8     | 23  | 25  |
| J7         | 32  | 33  | J9     | 25  | 28  |
| J9         | 35  | 40  | J8     | 28  | 29  |
| J10        | 42  | 44  | J10    | 29  | 31  |

The Table 4 shows the proposed ORR approach of resource reservation. The job J3 and J8 in scenario 1 are assigned in a single slice. Hence the overhead on process suspension and resumption is reduced. Similarly in scenario 2, the Job J4 is allotted as a single slice and J8 as two slices.

Table 4. The proposed ORR approach

| SCENARIO 1 | SCENARIO 2 |
|------------|------------|
| Allot      | ST  | ET  | Allot  | ST  | ET  |
| J1         | 3   | 7   | J1     | 1   | 4   |
| J2         | 9   | 12  | J2     | 4   | 6   |
| J3         | 12  | 15  | J3     | 8   | 10  |
| J4         | 15  | 19  | J4     | 12  | 15  |
| J5         | 22  | 24  | J5     | 15  | 18  |
| J8         | 24  | 27  | J6     | 18  | 19  |
| J6         | 27  | 30  | J7     | 20  | 23  |
| J7         | 32  | 33  | J8     | 23  | 25  |
| J9         | 35  | 40  | J9     | 25  | 28  |
| J10        | 42  | 44  | J10    | 29  | 31  |

4.1 Overhead on Process Suspension and Resumption

Among the many factors to be considered for process suspension and resumption, this paper considers the number of switches. Mostly the overhead on process suspension and resumption are machine dependent. Since there is denial of request in FCFS, there is no overhead incurred. In TARR, there were two process suspension and resumption whereas in ORR there is no process suspension and resumption. This is shown in the Figure 4.
5. Conclusion

The grid computing can be applied in around various organizations. Various researches are going on in increasing the efficiency of the organizations as it can be applied in educational institutions, in workflow workshop etc. In this paper, the advance resource reservation provides a guaranteed availability of resource, when it is required. There were various approaches available for ARR as reservation based on negotiation, priority, FCFS, Time Slice etc. But the proposed ORR approach considers the overhead incurred on process suspension and resuming. By series of operations it has been found that the overhead has decreased to a greater extent.

6. References

1. Buyya R, Murshed M. GridSim: A Toolkit for the modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. 2002;1–37.
2. Devi S N, Pethalakshmi. An Object Oriented Approach to Disseminate Education by Applying Grid Computing Techniques in Universities and Educational Institutions in India. Proceedings of 2012 IEEE International Conference on Technology Enhanced Education (ICTEE). INDIA: 2012.
3. Barzegar B, Esmaelzadeh H, Shirgahi H A New Method on Resource Management in Grid Computing Systems Based on QoS and Semantics. Indian Journal of Science and Technology. 2011 Nov; 4(11):1–4.
4. Venugopal S, Chu X, Buyya R K. A Negotiation Mechanism for Advance Resource Reservation Using the Alternate Offers Protocol. Proceedings of IEEE International Workshop on Quality of Service. Netherlands: 2008. p. 40–9.
5. Min R, Maheswaran M. Scheduling Advance Reservations with Priorities in Grid Computing Systems. Proceedings of 2002 2nd IEEE / ACM International Symposium on Cluster Computing and the Grid. 2002. p. 266–8.
6. Sulistio A, Buyya RK. A Grid Simulation Infrastructure supporting Advance Reservation, proceedings of the 16th International Conference on parallel and Distributed Computing and Systems. MIT, Cambridge, USA. 2004 Nov; 9 – 11:1–7.
7. Devi S N, Pethalakshmi A. TARR: Time-Slice based Advance Resource Reservation in Grid Computing Environments. International Journal of Computational Intelligence and Informatics. 2016 Mar.
8. Gomes E, Dantas M AR. Towards a resource reservation approach for an Opportunistic Computing Environment, Journal of Physics: Conference Series. 2014; 540:1–9.
9. Translation look aside buffer - Wikipedia, the encyclopedia. 2016 March 18. Available from: https://en.m.wikipedia.org/wiki/Translation_lookaside_buffer Tavakkolai H, Hosseinabadi AAR, Yadollahi M, Mohammadpour T. Using Gravitational Search Algorithm for in Advance Reservation of Resources in Solving the Scheduling Problem of Works in Workflow Workshop Environment. Indian Journal of Science and Technology. 2015 June; 8(11):1–16.