TARR: Time Slice based Advance Resource Reservation in Grid Computing Environments

S. Nirmala Devi
Department of Computer Science
Government Arts College for Women
Salem, Tamilnadu
devi.dharu@gmail.com

A. Pethalakshmi
Department of Computer Science
MVM Government Arts College for Women
Dindigul, India

***Abstract*- Grid computing provides a promising environment for the accomplishment of a particular task by sharing the resources when required. Grid environment is highly dynamic and also heterogeneous. The resources which normally shared are processor, storage and network bandwidth. Discovering the resource which suits the requirement itself is a tedious task. Sometimes, even after finding the resource, it may not be available as it may be used by some other task. To get the resource at the required time the resource can be reserved in advance. There are various advance resource reservation schemes as FCFS, Alternate Offer Protocol, priority based reservation etc. In this paper a new reservation scheme called TARR Time-Slice based Advance Resource Reservation is proposed. In this scheme, the reservation is done when the resource is free. If the resource is already reserved during that timeslot then the free the time slices can be used for the reservation. This splits the resource utilization period, i.e., whenever a free time-slice is available the resource is reserved for that duration and the remaining is deferred over a period of time where the free time slice is available. By applying this approach the Average Waiting Time (AWT) of the job to be completed decreases, the Hit-Ratio increases for fetching the resources and even the Resource Idle Time (RIT) decreases.**

Keywords- Grid Computing, Resource Management, Advance Resource Reservation, Time-Slice based Approach.

I. INTRODUCTION

Grid Computing may be defined as an agreement among the virtual organizations to accomplish the task assigned to the environment by sharing the resources. The task can be a computational task, storage task or service task. Based on the tasks the grid is classified as computational grid, data grid and service grid. The grid computing environment is dynamic where the computing nodes or the resources may enter or leave the environment at any time. The environment is also heterogeneous where each resource is of different type, capacity and from different vendors.

In order to make use of the grid efficiently, the resources in the grid environment can be shared to complete the task assigned to it. The required resources for the accomplishment of the tasks are required to be fetched from the environment. There are various approaches available to discover the resources as centralized approach, distributed approach, agent- based approach etc.,

Even after finding the required resource, it is not possible to make use of the resource, as the resource may be in use. Hence the resources can be reserved in advance for future use. The Advance Resource Reservation (ARR) scheme takes the Start Time (ST) and Finish Time (FT) of the resource utilization as the input and allots the reservation based on it. There are various advance resource reservation schemes available as FCFS (First Come First Serve) reservation based on alternate offers protocol, priority based reservation. There are certain drawbacks in the existing approaches.

In all existing approaches, the resources are reserved only when the resource is idle during the time slot specified in the start time and finish time. In the proposed approach the resources are reserved even in the time slices where the resource is kept idle after the start time of the reservation. This time slice reservation can be done until the Defer Time (DT) accepted by the user for a particular reservation. Moreover an upper limit is imposed on the number of time slice (NTS), to reduce the overhead on the resuming and suspension of the job due to the resource availability. By employing the proposed approach the AWT Average Waiting Time (AWT)

of the task to be completed decreases, the Hit Ratio (HR) of the resource reservation increases and also the Resource Idle Time (RIT) decreases.

In this paper the section II deals with the schemes available for advance resource reservation. In Section III, the proposed algorithm is given. Section IV deals with various metrics considered to analyze the performance the grid computing environment. Finally in section V the results are compared.

II. RELATED WORK

ARR in grid computing environment provides a guaranteed availability of the resources for the task to be completed. In this advance resource reservation topic, various researches are going on. All the reservation schemes take the ST and FT of the resource as the input to make the reservation.

Srikumaret. al.,[1] proposes the negotiation mechanism between the proposer and the responder. 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 is no time-limits or any constraints imposed in the protocol.

Rui Min et. al., [2] proposes reservation based on the priority 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.

In [3], Eliza et.al., proposes the resource reservation for an opportunistic computing environment. 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.

Anthony Sulistio et.al.[4] proposes the advance resource reservation based on the First Come First Serve (FCFS) approach. When a resource is requested for reservation then the reservation can be done on the 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 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 TARR approach tries to remove this drawback by allowing the usage of time-slices between the existing reservations

III. PROPOSED WORK

The Start, finish and defer time for using the resources are given by the user while submitting a job for execution. The start time and finish time are the expected start and finish time for using the resources. Sometimes it is not possible to reserve the resource at the stipulated start and finish time. In such case the reservation can be done until the defer time. Defer time is the time until which the reservation can be postponed.

A. The proposed TARR Algorithm

Algorithm TARR

1. if List is empty then
2. no conflict found. Hence accept the new reservation.
3. else
4. for i = 0 to List-Size -1 do
5. Put i into templist if one of the properties are true
6. $Start_{new} \leq start_i \&\& finish_{new} \geq finish_i \;||\; Start_{new} \leq finish_i || \; finish_{new} \leq finish_i$
7. end for

8. if templist is empty then
9. no conflict found. Hence accept this reservation.
10. else
11. requiredtime = $finish_{new}$– $start_{new}$
12. balancereserve = requiredtime
13. for i = 1 to n
14. timeslice = $start_{i+1}$ -$finish_i$
15. if timeslice> 0 then
16. Put into slicequeue_insert(id,sst,sft)
17. end if
18. end for
19. while (slicequeue&&balancereserve)
20. if sst ≥ $Start_{new}$ then
21. assign new reserve
22. currentreserve=sft – sst
23. Balancereserve = requiredtime –currentreserve
24. end if
25. end while
26. end if
27. end if

B. Reservation List

A reservation list is maintained at each resource. The list consists of the reservation id, ST and FT of the reservation. When there is no reservation for the resource then the list is empty. The new reservations can be added immediately when the list is empty. It is given in line 1 to 2 in the algorithm. If the reservation list is not empty then it is required to find out whether the required slot is free. For this the $start_{new}$ and $finish_{new}$ are compared with the existing list. In table 1 first scenario, J1 comes first and the reservation is made.

C. Finding the availability for new reservation

The start and finish time of the new reservation is checked with the existing one. If $start_{new}$ is less than the start of the any of the reservations and the $finish_{new}$ is greater than the finish of the any of the reservation then it means there is a conflict. Even if $start_{new}$ or $finish_{new}$ is less than finish of any of the available reservations then also there exists the conflict. Only when there is no conflict there is slot available for reservation. Hence the reservation can be made when there is no conflict.

D. Forming SliceQueue

When reservations are not made possible within the slots available, then the empty time slices are arrived and placed into the SliceQueue. For this the difference between the current finish time and the next start time is computed as TimeSlice in the equation (1).

$$TimeSlice = Finish_{current} - Start_{next} \quad (1)$$

These timeslices are placed into the SliceQueue with an id and the $finish_{current}$ and $start_{next}$ values. The $finish_{current}$ and the $start_{new}$ are stored as SST(Slice Start time) and SFT (Slice Finish Time).

E. Availing timeslices from SliceQueue

When there is reservation required in time slices for a job, then it can be provided. The balancereserve holds the expected reservation at that particular time. As and when a time slice is reserved then the time slice duration is reduced from the balancereserve which is computed using the equation (2).

$$balancereserve = requiredtime - currentreserve \quad (2)$$

wherecurrentreserve = SFT – SST. The reservation is done until the balancereserve becomes null.

IV. METRICS USED

The FCFS algorithm and TARR algorithm are implemented in JAVA. The metrics considered for comparison of the two algorithms are the Average Waiting Time(AWT), Hit-Ratio (HR) and the Resource Idle Time (RIT).

A. Average Waiting Time(AWT)

The waiting time (WT) of the reservations are computed. Sometimes the resources are not available at the time of reservation requirement. But the resources can be reserved within the deferred time. In that case, the difference between the expected start time and the actual reserved start time is the waiting time as in the equation (3).

$$\text{Waiting Time (WT)} = \text{Start}_{\text{reserve}} - \text{Start}_{\text{new}} \quad (3)$$

The Total Waiting Time (TWT) is computed as the sum of all the waiting time at a specific point of time as in the equation (4).

$$\text{Total Waiting Time (TWT)} = \sum_{i=1}^{\text{size}} \text{WT} \quad (4)$$

where the size refers to the length of the reservation list at a specific point of time. Then Average Waiting Time is computed using the equation (5)

$$Average\ Waiting\ Time(AWT) = \frac{Total\ Waiting\ Time\ (TWT)}{No\ of\ Reservations} \quad (5)$$

The AWT decreases while applying TARR reservation.

B. Hit Ratio (HR)

Hit Ratio refers to the number of reservations accepted. At the time of requesting resources certain resources may not be available for reservation, hence the request miss would happen. The hit ratio can be computed by the formula in equation (6)

$$\text{Hit Ratio (HR)} = \text{Number of Hit Reservation} : \text{Total Number of Reservations} \quad (6)$$

The HR increases in case of TARR when compared with FCFS as the reservation is granted even on time slice available.

C. Resource Idle Time (RIT)

The resources may be idle even when the reservation request available. This happens when the idle time does not fit into the allocation policy. Thus TARR provides a better allocation policy, as even the time-slices are used for reservation rather than allocating the entire request as a single unit. The RIT is computed by applying the below formula

$$\text{RIT} = \text{Finish}_{\text{previous}} - \text{start}_{\text{current}} \quad (7)$$

when there exist a reservation request with a conflict. The total resource idle time is computed by the following equation

$$Total\ RIT\ (TRIT) = \sum_{i=1}^{size} RIT \quad (8)$$

The RIT also decreases in case of the TARR.

V. PERFORMANCE ANALYSIS

The start time (ST), finish time (FT) and defer time (DT) for 10 jobs under 4 scenarios are listed in table 1. The table 2 depicts the reservation based on the FCFS model with its waiting time and Resource idle time. In Scenario1, the total waiting time is 13 and the RIT is 13. Table 3 depicts the TARR based resource reservation and it is observed that for the same scenario 1 the total waiting time is 6 and the RIT is 0. J8 is denied as the defer time by which the reservation is completed is less than the possible allocation time. The

comparison between the FCFS and TARR are shown in table 4.Similarly the waiting time, idle time are calculated for other scenarios and presented in table 2, table 3 and table 4.

Table 1. The four different scenarios seeking reservation

Scenario 1				Scenario 2				Scenario 3				Scenario 4			
JID	**ST**	**FT**	**DT**	**JID**	**ST**	**FT**	**DT**	**JID**	**ST**	**FT**	**DT**	**JID**	**ST**	**FT**	**DT**
J1	3	7	20	J1	1	4	15	J1	2	5	15	J1	1	3	10
J2	9	12	20	J2	4	6	25	J2	5	9	20	J2	4	7	15
J3	6	9	25	J3	8	10	20	J3	10	15	25	J3	2	4	20
J4	13	19	25	J4	5	8	25	J4	8	10	25	J4	8	10	25
J5	22	24	30	J5	12	15	25	J5	17	19	30	J5	12	16	30
J6	27	30	35	J6	18	19	30	J6	20	25	35	J6	18	21	35
J7	32	33	40	J7	20	23	35	J7	26	28	40	J7	17	19	40
J8	20	24	30	J8	21	25	40	J8	21	24	40	J8	23	26	40
J9	35	40	45	J9	25	28	40	J9	30	33	45	J9	27	30	45
J10	42	44	50	J10	29	31	40	J10	35	37	50	J10	30	32	45

Table 2. FCFS based resource reservation with waiting time and Resource idle time

Scenario 1						Scenario 2						Scenario 3						Scenario 4					
Allot	**ST**	**ET**	**UT**	**WT**	**RIT**	**Allot**	**ST**	**ET**	**UT**	**WT**	**RIT**	**Allot**	**ST**	**ET**	**UT**	**WT**	**RIT**	**Allot**	**ST**	**ET**	**UT**	**WT**	**RIT**
J1	3	7	4	0	0	J1	1	4	3	0	0	J1	2	5	3	0	0	J1	1	3	2	0	0
J2	9	12	3	0	2	J2	4	6	2	0	0	J2	5	9	4	0	0	J2	4	7	3	0	1
J4	13	19	6	0	1	J3	8	10	2	0	2	J3	10	15	5	0	1	J4	8	10	2	0	1
J3	19	22	5	13	0	J5	12	15	3	0	2	J4	15	17	2	7	0	J3	10	12	2	8	0
J5	22	24	2	0	0	J4	15	18	3	10	0	J5	17	19	2	0	0	J5	12	16	4	0	0
J6	27	30	3	0	3	J6	18	19	1	0	0	J6	20	25	5	0	0	J6	18	21	3	0	1
J7	32	33	1	0	2	J7	20	23	3	0	0	J7	26	28	2	0	1	J7	21	23	2	4	0
J9	35	40	5	0	3	J9	25	28	3	0	2	J9	30	33	3	0	2	J8	23	26	3	0	0
J10	42	44	2	0	2	J10	29	31	2	0	1	J10	35	37	4	0	2	J9	27	30	3	0	0
						J8	31	36	5	10	0	J8	37	40	3	16	0	J10	30	32	2	0	0

Table 3. TARR based resource reservation with waiting time and Resource idle time

Scenario 1						Scenario 2						Scenario 3						Scenario 4					
Allot	**ST**	**ET**	**UT**	**WT**	**RIT**	**Allot**	**ST**	**ET**	**UT**	**WT**	**RIT**	**Allot**	**ST**	**ET**	**UT**	**WT**	**RIT**	**Allot**	**ST**	**ET**	**UT**	**WT**	**RIT**
J1	3	7	4	0	0	J1	1	4	3	0	0	J1	2	5	3	0	0	J1	1	3	2	0	0
J3	7	9	2	1	0	J2	4	6	2	0	0	J2	5	9	4	0	0	J3	3	4	1	0	0
J2	9	12	3	0	0	J4	6	8	2	1	0	J4	9	10	1	1	0	J2	4	7	3	0	0
J3	12	13	1	3	0	J3	8	10	2	0	0	J3	10	15	5	0	0	J3	7	8	1	0	0
J4	13	19	6	0	0	J4	10	11	1	2	0	J4	15	16	1	5	0	J4	8	10	2	0	0
J8	20	22	2	0	0	J5	12	15	3	0	0	J5	17	19	2	0	0	J5	12	16	4	0	0
J5	22	24	2	0	0	J6	18	19	1	0	0	J6	20	25	5	0	0	J7	17	18	1	0	0
J8	24	26	2	2	0	J7	20	23	3	0	0	J8	25	26	1	4	0	J6	18	21	3	0	0
J6	27	30	3	0	0	J8	23	25	2	2	0	J7	26	28	2	0	0	J7	21	22	1	3	0
J7	32	33	1	0	0	J9	25	28	3	0	0	J8	28	30	2	2	0	J8	23	26	3	0	0
J9	35	40	5	0	0	J8	28	29	1	3	0	J9	30	33	3	0	0	J9	27	30	3	0	0
J10	42	44	2	0	0	J10	29	31	2	0	0	J10	35	37	2	0	0	J10	30	32	2	0	0
						J8	31	32	1	2	0												

Table 4. Comparison between the FCFS and TARR

Scenario	AWT		HR		TRIT	
	FCFS	TARR	FCFS	TARR	FCFS	TARR
Scenario 1	1.3	0.6	9:10	1:1	13	0
Scenario 2	2	1	1:1	1:1	7	0
Scenario 3	2.3	1.2	1:1	1:1	6	0
Scenario 4	1.2	0.3	1:1	1:1	3	0

After analyzing the scenarios it is found that the Average Waiting Time (AWT) is lower for all the cases. Only when the entire slot is available then it is allotted in FCFS. Whereas in TARR the allocation is made in time slice when there are no avenues for allotment as a whole. The figure 1 depicts the AWT in both FCFS and TARR.

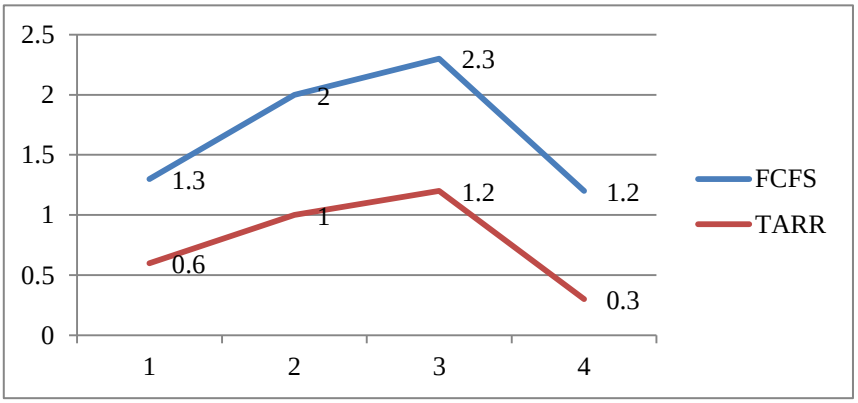

Figure 1: AWT for FCFS and TARR

While comparing the resource idle time TARR always provides at most resource utilization. There is resource idle time in FCFS. This is graphically represented in figure 2. Because of time slice approach there is no resource idle time in TARR.

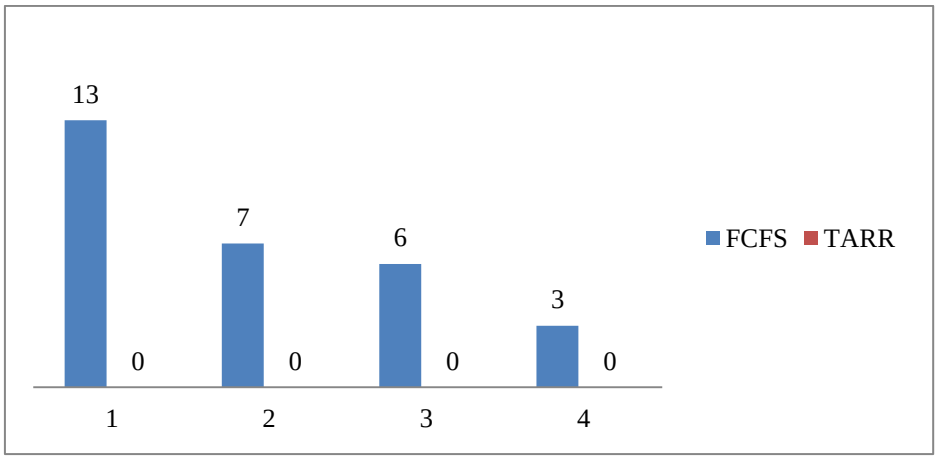

Figure 2: Resource Idle Time in FCFS and TARR

VI. Conclusion

Advance resource reservation assures the availability of resources when required, which enhances the efficient utilization of resources and also reduces the execution time of a process. There were various approaches available for advance resource reservation but those approaches strive hard to provide the resource as a whole slot. As a consequence, there can be only two outcomes as either to provide the resource in full slot or reject the reservation. In the proposed approach, the time slices are granted in slots. The defer time until which reservation can be made is obtained from the user. The resource idle time slots until the defer time is obtained and they are allotted. Hence this reduces the resource idle time. This proposed approach even reserves the resource in time slots hence it reduces the waiting time of the processes, which in turn reduces the overall average waiting time of the processes.

References

[1] SrikumarVenugopal, Xingchen Chu and RajkumarBuyya, "A Negotiation Mechanism for Advance Resource Reservation Using the Alternate Offers Protocol". In the proceedings of 2008 16th IEEE International Workshop on Quality of Service, Netherlands,40 – 49,2008.

[2] Rui Min and MuthucumaruMaheswaran, "Scheduling Advance Reservations with Priorities in Grid Computing Systems". In proceedings of 2002 2nd IEEE / ACM International Symposium on Cluster Computing and the Grid, 266 – 268, 2002.

[3] Eliza Gomes and M.A.R.Dantas, "Towards a resource reservation approach for an Opportunistic Computing Environment", Journal of Physics: Conference Series 540, 2014.
[4] Anthony Sulistio and RajkumarBuyya, "A Grid Simulation Infrastructure supporting Advance Reservation", Proceedings of the 16thIntenational Conference on parallel and Distributed Computing and Systems, MIT, Cambridge, USA, 1-7, 2004.

BIOGRAPHY

S. Nirmala Devi was born in Erode, Tamil Nadu (TN), India, in 1976. She received the Bachelor of Commerce (B.Com)degree from the University of Madras, Chennai, TN, India, in 1996 and the Master of Computer Applications (M.C.A.) degree from the University of Madras, Chennai, TN, India, in 1999. She is currently pursuing the Ph.D. degree with the Department of Computer Science, Manonmaniam Sundaranar University, Tirunelveli. Her research interests include Grid Computing and Data Mining.

A. Pethalakshmi Annamalai received Master of Computer Science (M.Sc) degree from the Alagappa University, Karaikudi, TN, India in 1988 and received Master of Philosophy in Computer Science From Mother Teresa Women's University, Kodaikanal, TN, India in 2000. She has received her Ph.D Degree from the Department of Computer Science, Mother Teresa Women's University, Kodaikanal, TN, India in 2008. Currently she is working as Associate Professor and Head, Department of Computer Science. MVM Govt. Arts College (w), Dindigul, TN, India. Her area of interests include fuzzy, rough set, neural network and grid computing.