International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 3, March - 2013

A Comparative Study Of Advanced Reservation Algorithms In Optical Grid

[1]Mahesh A. Khandke, [2]Rajkumar B. Pawar, [3]Anil R. Chinchawade
Department of Computer Engineering,
Gharda Institute of Technology
Lavel, Ratnagiri, M.S., India

Abstract

The ability to conduct advance reservations within optical grid environments is crucial for applications that want to utilize distributed resources in a efficient manner. Advance reservations are essential for supporting scheduling of distributed resources. Further, advance reservations can significantly enhance the capabilities of resource brokers. Four different algorithms for advanced reservations such as Decomposion Algorithm, Greedy Algorithm, ESnet On-Demand Secure Circuits and Advanced Reservation System (OSCARS), Gravitational Emulation Local Search Advanced Reservation Algorithm (GELSAR) are studied here. Also their comparison has been done depending on the parameters such as quality, complexity and so on.

Key words: Advance reservation, optical grid

I. Introduction:

Advanced Reservation is a contract between the resource owner and consumer that commits a certain resource for a defined time to the resource consumer. It can ensure the future availability of the Grids heterogeneous resources and help a scheduler to produce better schedules [4].
With advance reservations users can obtain execution guarantees from local resource managers without requiring detailed knowledge of current and future workloads or of the resource owner's policies. This mechanism guarantees the availability of resources to users at some specified future time. Since the resources are limited per time unit, the provider may be unable to meet all demands, so the broker must choose which requests are to be accepted. The idea here is study the algorithms to maximize the utility of the broker by selecting an optimal subset of customer's orders.
Customer orders use the following reservation pattern:

- Start and end time- which gives starting & ending time
- Requested Quality of Service (QoS), i.e., amount of resources per time unit
- Price proposed for the service
- Penalty if the QoS is not satisfied

II. Algorithms for Advanced Reservation

II.1 Decomposition Algorithm

Here author's of [4] represent the set accept by labeling each order (or order) with accept or reject. In this algorithm, first solver uses branching on accept/reject alternatives combined with a decomposition strategy which breaks the original problem in independent sub problems. It applies various operations to perform its decomposition. The operation are as follows:

1. RemoveTime -which is performed each time the capacity exceeds the demand.
2. ForcedAccept- which can force the acceptance of the third order. We apply it each time the required QoS is compatible with the resource capacity.
3. ForcedReject - when the demand is higher than capacity.
4. Split operation- breaks the problem in two independent sub problems [4].

The decomposition algorithm which applies the previous operations is shown in Algorithm 2. It starts with the preprocessing defined in Algorithm 1.

```
Algorithm 1: Pre-processing
Input: P, an advanced reservation problem;
begin
Reduce1(P);
Split1(P) into set of problems S;
for (s in S) do
Solve(s);
end
```

After performing preprocessing, the algorithm successively applies two operations.
namely rejection of some order o & after rejection, Reduce2 operation on the problem. This will result in a simplified problem which may be split into independent sub-problems S. On each sub-problem algorithm is recursively applied.

```
Algorithm 2: The Decomposition Algorithm
Input: P, an advanced reservation problem;
begin
if orders = $\Phi$; then return;
else
Select an order o from orders;
Non  deterministically do one of;
(1)RejectOrder(o);
Reduce2(P);
Split(P) into set of problems S;
for s in S do Solve(s)
(2)AcceptOrder(o);
Reduce3(P);
Split(P) into set of problems S;
for s in S do Solve(s)
end
```

II.1.1 Refining the Algorithm

Now refine the decomposition algorithm by defining the reduce and split operations. In what follows let demand(t : times) be :QoS(o)

Also let next(t) be the smallest time strictly greater than t; next(t) is undefined if t is the largest time. In defining reduce and split operations we will make use of the following operations:

- RejectOrder(o:orders): Remove order from list as it requires QoS than the capacity.
- AcceptOrder(o:orders): Accept that order & removes it from waiting list. Also subtract the QoS of that order o from capacity.
- TestRemoveTime(t:times): if demand(t) is less than or equal to capacity(t) then RemoveTime(t)
- TestForcedReject(o:orders): if for some t in duration(o), QoS(o) > capacity(t) then RejectOrder(o)

II.2 Greedy Algorithm

Here author's of [4] has given second algorithm which performs a greedy selection of the orders. This algorithm can use various strategies to select orders. Author has defined greedy algorithmas follows:

if o is an order then rate(o) is price(o)/(QoS(o)*duration(o))

```
1 Algorithm 3: The Greedy algorithm
2 Input: P, an advanced reservation problem;
3 begin
4 for t = earliest time to latest time do
5 while capacity(t) < demand(t) do
6 let o be a minimal rate order from {o'|t ε duration (o')};
7 RejectOrder(o);
8 Let R be the set of all orders labelled reject;
9 for o ε R, generated from highest to lowest rate do
10 if  for all t ε duration(o); capacity(t) - demand(t) > QoS(t) then
11 unlabel o and put o in orders;
12 Accept all orders;
13 end
```

The algorithm starts with the processing of conflicting situation, i.e., when the demand exceed the capacity. Each time it finds such a situation, it rejects some minimal rate order. After this initial filtering, the algorithm considers rejected orders. Each time some rejected orders has a QoS requirement compatible with the remaining capacity, the order is unlabeled and put in the initial set of orders. After this, all the unlabeled orders can be accepted [4].

II.3 The ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS)

The Energy Sciences Network (ESnet) provides high bandwidth connections between research laboratories and academic institutions for data sharing and video/voice communication. The ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS) establishes guaranteed bandwidth of secure virtual circuits at a certain time, for a certain bandwidth and length of time. Though OSCARS operates within the ESnet, it also supplies end-to-end provisioning between multiple autonomous network domains. OSCARS gets reservation requests through a standard web service interface, and conducts a Quality of service (Quos) path for bandwidth guarantees [5].

Bandwidth reservation system is used for keeping track of changes in the network status in OSCARS. With help of that status a topology graph G is maintained as follows:

Every port in a router has a maximum bandwidth available for reservation, and each network link connecting two ports (providing communication from one router towards another one) has an engineering metric related to the link latency. The web service interface enables users to allocate a fixed amount of bandwidth for a time period between two end-points in the network. A reservation request R contains: source and destination end-points, requested bandwidth, and the start/end times, R=nsource, ndestination, Mbandwidth, tstart, tend [5].
The reservation engine needs to ensure availability of the requested bandwidth from source to destination for the requested time interval, because there might be bandwidth guaranteed paths in the system that are already fully or partially committed.

To avoid ambiguity of the committed reservations all the reservations in between tstart, tend are examined. From this examination a snapshot graph G' of the network topology is generated. G'=G(tstart, tend) represents status of the network in advance. Based on the engineering metric on each link shortest path on G'=G (tstart, tend) from source to destination is calculated. From this G' a bandwidth guaranteed path is set up to commit and eventually complete the reservation request for the given time period.

If the requested reservation cannot be granted, a failure message will be generated. Therefore in this algorithm user has to use a trial-and-error sequence for a particular reservation. That is user does not have optimal choice. So to avoid that, authors of [5] has enhanced the OSCARS reservation system by extending the underlying mechanism to provide a new service in which users submit their constraints and the system suggests possible reservation requests satisfying users' requirements.

In OSCARS algorithm, users are providing information such as provide maximum bandwidth they can use, total data size requested for transmission, the earliest start time, and the latest completion time. Also the users can set criteria such that they would like to reserve a path for earliest completion time or reserve a path for shortest transfer duration. Such a request can be represented as: Rs'=nsource, ndestination, MMAXbandwidth, DdataSize, tEarliestStart, tLatestEnd [5].
On capability of the client and server hosts between source and destination end-points, maximum bandwidth is depends. The reservation engine finds out the reservation R=nsource, ndestination, Mbandwidth, tstart, tend for the earliest completion or for the shortest duration where Mbandwidth $\leq$ MMAXbandwidth and tEarliestStart $\leq$ tstart $<$ tend = tLatestEnd.

The outline of OSCARS approach presented in [5] is as follows. Here the given search interval is divided into several time windows, and keep snapshots of the network topology about the available bandwidth status for every link in each time window. This information is updated on-the-fly every time a reservation request is committed and stored for further processing during the path calculation phase.

A time window represents a period of time in which we have a stable discrete status in terms of available bandwidth over the links. For example, if we have three committed reservations with allocated bandwidth for their time periods r1=b1, t1, t3, r2=b2, t2, t5, r3=b3, t2, t4, where the times t1, t2, t3, t4, t5 are distinct values, there will be four time windows tw1=t1, t2, tw2=t2, t3, tw3=t3, t4, tw4=t4,t5 and four snapshots for the time windows Gtw1, Gtw2, Gtw3, Gtw4. If a link is associated with all three paths in these three reservations r1, r2, r3, then, the available bandwidth over that link is equal to bandwidthmax (b1+b2+b3) for the time period of (t2, t3), which is kept in Gtw2. The next step is to search through these time windows in a sequential order to check whether we can satisfy the requested allocation for that time window. For the given example above, first tw1 (t1, t2), and tw2 (t2, t3) will be examined; later, if both cannot satisfy the request, time window tw12, a combination of tw1 and tw2 (t1, t3), will be examined. This can easily be computed using Gtw1 and Gtw2 such that Gtw1-2=bbandwidth (link i) = min(bbandwidtht(Gtw1(linki), bbandwidth(Gtw2(linki)). For earliest completion time, the search pattern will be as follows: tw1, tw2, tw1-2, tw3, tw2-3, tw1-3, tw4, tw3-4, tw2-4, and tw1-4. The additive property of Gtw makes the process easy, since we only need to store one graph snapshot for each starting time window; for example, to obtain Gtw1-4we only need Gtw1-3 and Gtw4.

II.4 Gravitational Emulation Local Search Advanced Reservation Algorithm (GELSAR)

In [6], authors have described GELSAR algorithm. In this algorithm gravity law is used to solve advanced reservation problems. In this method, as an initial solution computational resources of Grid system are allocated to the users' requests. Consider the scheduler S for the grid system with N server which is distributed in the network geographically. The authors [6] have considered that the processing capacity of all servers is identical and is equal to C. A user with job j offers the service request to the scheduler. This request can be specified with three parameters [rj, lj, dj] where:

- rj: is the ready time of the job, the earliest the job can be made available to the grid system for processing
- Lj: is the length of the job, the amount of work the job requires
- dj: (lj + rj <= dj) : is the deadline of the job, the latest time for the job to be completed

If scheduler S determines that the schedule cannot be completed within given deadline, the given job is discarded and its user accordingly.

For each server i the time periods in future during which the server is reserved for a jobs (advanced reservation) scheduler S maintain a schedule. In fact, this schedule represents the set of advance reservations that have been made, and it guarantees that server resources will be available to the accepted jobs at specific future time.
Suppose that at same time, several jobs with the preparation time 1 and variant run time, a request service will be asked. When a service request [rj, lj, dj] is simultaneously given to several new jobs, the scheduling S immediately runs an algorithm to determine whether scheduling S uses a set of criteria and standards to select one of the servers who can handle this job, otherwise it drops the job. The Scheduling algorithms will increase the percentage of accepted jobs and the system utilization.

II.4.1 Gravitational Emulation Local Search algorithm (GELS)

In 1995, Voudouris and Tesang offered the algorithm GEL for searching in a search space and solving the example NP-complete for the first time, and in 2004 Barry and Webster [1] offered the algorithm as a powerful algorithm and it was called GELS. This algorithm introduced randomization concept along with two of four primary parameters i.e. velocity and gravity in physics through swapping in terms of groups by using random number in the existing local search algorithm GELS in order to avoid local minima [6].

GELS take as its basis the natural principles of gravitational attraction. Gravity works in nature to cause objects to be pulled towards each other. The more massive the object, the more gravitational tug it exerts on other objects. Also, the closer two objects are to each other, the stronger the gravitational forces between them. This means that a given object will be more strongly attracted to a larger, more massive object than to another object of lesser mass at a given distance, and it will also be more strongly attracted to an object close by than to another, more distant object having the same mass [6].

Processes of nature for searching in a Search space can be emulated by GELS. The idea is to imagine the search space as being the universe and object in this universe are the possible solution for the search. The mass of the solution is depends on its objective function value. If the solutions objective function value is better, then its mass is higher. A zero mass is assigned to locations within the search space that do not contain valid solution.

In this method, the possible solutions in the search space divide into some sets based on criteria. These criteria depend on the kind of problem and each of these sets is called a dimension of the problem solution. For each dimension of the problem solution a value entitled initial velocity has been intended, that it will be explained in continuation [6].

GELS use two methods for computing the gravitational force between the solution or the objects in the search space. The first method, a solution from the local neighborhood space is selected as a current solution and the gravitational force between these two solutions can be computed. The second method applies the formula to all solution within the neighborhood and tracks the gravitational force between each of them and the current solution individually all solution. In the movement through the search space, GELS acts in two modes too. The first mode, allows movement only to solutions within the current local search neighborhood. Each one of these movement modes can be used with each of the computation gravity force and as a result those four models GELS are made [6].

GELS maintain a vector, for representing relative velocity in each dimension. Depending on the number of dimensions in a solution size of the vector is determined. The algorithm is as follows [6]:

1. Initialize the current solution, velocity vector and direction of movement.
2. For each dimension in the velocity vector,
   i. A random integer between one and the maximum velocity is chosen, and this becomes the value of the element at that dimension.
3. The initial solution can be made as current solution either with user or randomized.

4. For each dimension in the initial velocity vector
   i. A direction is selected for the movement that the direction is equal to the solution dimension which has the largest value in the initial velocity vector

Algorithm includes a pointer object that can move through the search space and a mass which is intended for the pointer object is stable in the whole computations and this object refers to a solution with the largest mass. The algorithm will terminate when one of following two conditions occurs: either all of the element in the initial velocity vector have gone to zero, or the maximum allowable number of iterations has been completed. In each algorithm iteration as the first method a candidate solution will be selected from the local neighborhood space of the current solution based on the direction of the current movement and the gravity force between the current solution and the candidate solution is computed and then the velocity vector concerning this force will be update. For the next frequency, the velocity vector is checked and concerning it the movement direction can be chosen. Each iteration algorithm by the second method is completely as similar as the first method, but there is a little difference, as the gravity force and the initial velocity of the update action is computed for each of the candidate solutions instead of the gravity force computation and the update action of the velocity vector for only an obtained candidate solution from the current direction. Newton's formula is used, with the alteration that the two masses in the numerator of the equation are replaced by the value of the difference between the objective function value of the candidate solution and that of the current solution [6].

The value of the gravitational force between the two solutions then becomes:

$$f = G\,(CU-CA)/R2$$

In this formula, CU and CA are the value of the current solution and the candidate solution. If the objective function value of the current solution is larger than that of the candidate solution then the formula is giving positive values. If the candidate value is larger then it will give negative values. Then the value of this force, positive or negative, can be added to the velocity vector in direction of the current movement. If doing so makes the value exceed the maximum velocity parameter setting, it is set to the maximum. If the update would case the value to go negative, it is set to zero [6].

Some parameters which can be accessible are involving in:

- Maximum Velocity: act as maximum threshold value of the velocity above which all the velocities are unusable

- Radius: it is a radius that can be used in the formula of the gravitational force computation
- Iteration: it gives total number of iterations of the algorithm that will be allow to complete before process is terminated automatically

In the use of the above simulator in the design of the advanced reservation resources in Grid computational system has several conceptions in the algorithm GELSAR that they must be specified in this example that they are involved in [6]:

Definition Dimension Solution

In the GELSAR algorithm, the dimensions of the solution can be equally intended as the number of the offered jobs to the scheduler that must be reserved in Grid computational system. Dimension of the solution is nothing but each offered job to the scheduler. The current solution neighborhood in the selected dimension is equal to a solution where we are getting stable characteristics of the machine and or resources. Also they are getting it from neighborhood by changing the characteristics of the other machines or resources.

Definition Neighborhood

In the algorithm GELSAR as compared to the other search algorithms, the neighborhood solution is taken serially. Every current solution has different neighborhoods and each one is based on movement direction toward the neighborhood solution that is kind of particular change. Based on this neighborhood all neighborhoods are formed. For finding the neighborhood solution, the number of the allocated resource is changed depending on dimensions which cannot be performed by the allocated recourses.

Solution Problem

In this problem, resources such as bandwidth are reserved for future use by a group of jobs. With this advanced reservation system utilization can be increased. Each job should be only allocated to one resource and then reserve it. Each resource may allocate to zero or many jobs. By utilizing the Algorithm GELSAR with this condition, an appropriate reservation factor should be defined. The reservation factor is defined as the number of the reserved jobs for the resources in the future. The set of the Job assignments to the resources is called as a solution. This solution can be used as encoded array length n that is equal to the number of the jobs so as to represent a solution [6].

Each subscript of the array represents one of the resource numbers that the job will be done on it. As it was explained by algorithm GELSAR to initialize GELSAR, a starting solution is selected, along with experimental value of the adjustable operation parameters. Then the running of GELSAR commences, and when it is completed, the solution for each job shows the assigned reserved resource along with its reserve factor [6].

The GELSAR is implemented using software Dephi7. GELSAR has the following Parts and facilities [6]:

1. In the simulated algorithm, first the steps number, resources and jobs should be specified.
2. With a random producer, the free time of each resource and the run time of each job will be specified.
3. The velocity vector values are randomly selected from the gorge of the values 1 up to the total number of jobs.
4. In each step, the highest index of the velocity vector is selected and then the allocated resources to that dimension will be changed. In fact a new object is made and then the current solution value is computed. If the value of the current solution is greater than the best gained solution, it would be selected as the best object up to this step.

5. The gravitational force is computed based on the value of the current object solution and candidate.
6. Then they gained gravitational force in the step 5 will be added to the velocity vector in each index.

The steps 1 to 6 will be continued until one of the two conditions does not happen: The whole initial velocity vector elements should be equal to zero, or the number of the iteration algorithm arrives at its own number [6].

II.5 Analysis of the Advanced Reservation Algorithms

II.5.1 Decomposition Algorithm

The worst case time complexity of Decomposition algorithm is exponential on the size of the input as the advanced reservation problem is NP-hard. If we are solving the problem using branch-and-bound technique then we are getting average time complexity.

II.5.2 Greedy Algorithm

In this algorithm, arrays indexed by time slots are maintaining demand and capacity information. The initial state of the capacity array is given as an input. The initial state of the capacity array is given as an input. Constructing the initial state of the demand array requires iteration over |orders| orders. Authors of [4] have maintained an array of the set of orders active at a particular time. From that array orders are selected based on rate.
If we use a balanced binary search tree, construction takes O (|times|* *log* |*orders*|)
So*;* the first loop has worst case time complexity is *O*(|*times*| * |*orders*| * *log* |*orders*|).
The second loop is over the rejected set; |R| which is bounded by |orders|:
So; the overall time complexity is: *O* (|*times*| * |*orders*| * *log* |*orders*|).
Demand and capacity are represented through arrays of |times| elements.
The active array has size |times| *|*orders*|
The Greedy Algorithm returns only correct optimal solutions, because in this demand does not exceed capacity at any time. In the first stage set of all accepted orders whose capacity do not exceed maximum capacity at any time. Also, after addition of order in the set of demands, does not exceed capacity at any time.

II.5.3 The ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS)

The complexity of the OSCARS algorithm is as follows. The complexity of Max bandwidth path algorithm is O (N2), where N is the number of routers. Number of committed reservations within the given period of ($t_{EarliestStart}$, $t_{LatestEnd}$) decides searching of the window. In the worst-case, we may require to search all time window combinations, which is T(T+1)/2, where T is the number of time windows. If there are r committed reservations in that period, there can be maximum 2r+1 different time windows in worst-case. Overall, worst-case complexity is bounded by $O(r^2N^2)$. However, r is relatively very small compared to the number of nodes N, in the topology.

Authors of [5], has reserved bandwidth for data transfer. They have tested the performance of the algorithm by simulating very large graphs (with 10K nodes) and they have observed that computation time is in the order of seconds. Network provisioning is not sufficient by itself for end-to-end high performance data transfer. In order to take advantage of the available bandwidth, client sites should have storage allocation.

II.5.4 Gravitational Emulation Local Search Advanced Reservation Algorithm (GELSAR)

Gravitational Emulation Local Search Advanced Reservation algorithm or GELSAR is presented to handle scheduling and the advance reservation of the resources. In [6], the algorithm GELSAR is used several times and comparison has been done with the Genetic algorithm. In [6], authors have shown two algorithms with forty jobs and the number of the algorithm iteration 1000, the number of the allocated jobs. That test shows that the GELSAR algorithm has better execution time (around 50%) as compare to Genetic Algorithm. It also increases the jobs reservation factor up to 7/5 percent [6].

Table 2.1 compares these four advanced reservation algorithm.

	Decomposition Algorithm	Greedy Algorithm	OSCARS Algorithm	GELSAR Algorithm
Speed	Slow	Quick	very slow	Quick
Quality	Gives optimal solution	Gives nearly optimal solution	Cannot get optimal solution	Gives optimal solution
Complexity	Exponential on the size of input	O(|times|*|orders| **log* |orders|)	$O(r^2N^2)$	greater execution time

Table 2.1: Comparison of Advanced Reservation Algorithm.

III. Conclusion

Advanced reservation represents is an important mechanism in Optical Grid. It allows applications to request resources for use at a specific time in the future. For advanced reservation in Optical Grid many algorithms are present. In this paper we have compared algorithms such as Decomposition algorithm, greedy algorithm, OSCARS & GELSAR algorithm based on parameters such as speed, quality & complexity.

The comparisons shows that, execution speed of greedy algorithm & GELSAR algorithm is fast as compare to other two. Also the execution time shows the same. Decomposition algorithm, greedy algorithm & GELSAR give optimal solution while OSCARS fail to give optimal solution.

REFERENCES

[1] Ahmar Abbas, *Grid Computing: A Practical Guide To Technology and Applications*, Firewall Media, 2008.

[2] Ian Foster, Carl Kesselman, Steven Tuecke , “The Anatomy of the Grid", 2001.

[3] Ian Foster, Carl Kesselman, Steven Tuecke , Jeffrey M. Nick, “The Physiology of the Grid", 2002.

[4] Mark Bartlett, Alan M. Frisch, Youssef Hamadi, Ian Miguel, Chris Unsworth, “Efficient Algorithms for Selecting Advanced Reservations", IEEE,2006.

[5] Mehmet Balman, Evangelos Chaniotakis, Arie Shoshani, Alex Sim, “Advance Network Reservation and Provisioning for Science", July 2009.

[6] Behnam Barzegar, Amir Masoud Rahmani, Kamran zamani far, "Gravitational Emulation Local Search Algorithm for Advanced Reservation and Scheduling in Grid Systems", IEEE, 2009.

[7] Jawad Ashraf and Thomas Erlebach, “A New Resource Mapping Technique for Grid . . .. Workflows in Advance Reservation Environments", IEEE, 2010.

[8] Feng Liang, Shilong Ma, Andre Luckow, Bettina Schnor, “Earliest Start Time Estimation for Advance Reservation-based Resource Brokering Within Computational Grids", IEEE, 2010.

[9] Hidemoto Nakada, Atsuko Takefusa, Katsuhiko Ookubo, Makoto Kishimoto Tomohiro Kudoh, Yoshio Tanaka, Satoshi Sekiguchi, “Design and Implementation of a Local Scheduling System with Advance Reservation for Co-allocation on the Grid", IEEE, 2006.

[10] Savera Tanwir, Lina Battestilli, Harry Perros and Gigi Karmous-Edwards, "Dynamic Scheduling of Network Resources with Advance Reservations in Optical Grids ", 2007

[11] Min Zhu, Wei Guo, Shilin Xiao, Anne Wei, Yaohui Jin, Weisheng Hu, and Benoit Geller, "Availability-Driven Scheduling for Real-Time Directed Acyclic Graph Applications in Optical Grids", Opt. Comm. Network, 2010

[12] Nipatjakorn Kannasoot and Jason P. Jue, "Resource-Efficient Task Assignment and Scheduling in Optical Grids", 2010

[13] Wei Guo, Weiqiang Sun, Weisheng Hu, Yaohui Jin, "Resource Allocation Strategies for Data-Intensive Workflow-based applications in Optical Grids", IEEE, 2006