Genetic Algorithm based Data-aware Group Scheduling for Big Data Clouds

Raghavendra Kune[1], Pramod Kumar Konugurthi[1], Arun Agarwal[2], Raghavendra Rao Chillarige[2], and Rajkumar Buyya[3]
{raghav.es, pramodkumar.konugurthi, aruncs.2011}@gmail.com, vijaya_crr@yahoo.co.in, rbuyya@unimelb.edu.au

[1]Advanced Data Processing Research Institute, Department of Space, India
[2]School of Computer and Information Sciences, University of Hyderabad, India
[3]CLOUDS Lab, Department of Computing and Information Systems, University of Melbourne, Australia

Abstract- Cloud computing is a promising cost efficient service oriented computing platform in the fields of science, engineering, business and social networking for delivering the resources on demand. ***Big Data Clouds*** is a new generation data analytics platform using Cloud computing as a back end technologies, for information mining, knowledge discovery and decision making based on statistical and empirical tools. MapReduce scheduling models for Big Data computing operate in the cluster mode, where the data nodes are pre-configured with the computing facility for processing. These MapReduce models are based on compute push model- pushing the logic to the data node for analysis, which is primarily for minimizing or eliminating data migration overheads between computing resources and data nodes. Such models, however, substantially perform well in the cluster setups, but are infelicitous for the platforms having the decoupled data storage and computing resources. In this paper, we propose a Genetic Algorithm based scheduler for such Big Data Cloud where decoupled computational and data services are offered as services. The approach is based on evolutionary methods focussed on data dependencies, computational resources and effective utilization of bandwidth thus achieving higher throughputs.

Keywords: Big Data, Cloud computing, Data Intensive Scheduling, Genetic algorithms, Big Data Clouds.

I. INTRODUCTION

Big Data Cloud is an emerging data analytics platform for collecting, organizing and analyzing large data sets for discovering patterns and useful information. Big Data Cloud could be categorized into two types; one with enough computing resources at the data node, similar to a cluster setup, and the second with decoupled computational and data resources spread across several geographical locations. Big Data Analytics 0 are emerging data science paradigms for exploiting the large scale, multi-dimensional, and rapidly growing data for the intrinsic information extraction using computational and statistical methods. These analytics have wide spread applications in several fields like social networking analysis, business forecasting, financial domain analysis, scientific analysis etc. Big Data computing differs from traditional Data warehousing (OLTP/OLAP) technologies in terms of the data storage, organization, collection, processing tools and methods used for data analysis. Data warehousing deals with the operational data which is mostly structured, however, Big Data computing addresses both historical and operational data which is structured as well unstructured data.

MapReduce scheduling models [2] are primarily focussed on computing at the data nodes, such models are desirable for the applications that are demanding larger data volumes, but with minimal computing resources for processing. These techniques are attempted for collocated data and computing resources, however, may result in the degraded performance when the resources are decoupled. On the other side, traditional job scheduling approaches such as match making, optimizes either the computing time or the data migration overheads, however, both are not addressed together due to the limitations in such methods. Hence, MapReduce models and match making models are not tailored for scientific computing platforms where the resources are decoupled, and the requirement is to optimize both computing and data consolidation overheads. The adoption of this model is described in our earlier work for satellite data product generation [3]. To address such Big Data scientific workloads, we propose a scheduling methodology based on data grouping and optimizing mechanisms. The work in this paper is focussed on scheduling the jobs which are demanding multiple data sets that are spawned across several storage repositories, and the data demanded by such jobs are either similar or may have overlap in the data regions.

Genetic Algorithms (GA) [4][5] are the optimization techniques, used to solve NP class problems for finding an approximately optimal solutions. GA is a model of machine learning that derives the behaviour from a metaphor of the processes of evolution in nature. GA is executed iteratively on a set of coded chromosomes, called a population, with three basic genetic operators such as selection, crossover and mutation. Each member of the population is called a chromosome (or individual) that undergoes evolution carrying the fittest chromosomes to the next generations. This process is repeated until the specified maximum numbers of generations are reached or the optimal fittest value is obtained. In this paper, we describe a scheduling model based on GA and the model is evaluated and compared with earlier works such as match making and other heuristics techniques through the simulated data.

978-0-7695-5429-7/14 $31.00 © 2014 IEEE
DOI 10.1109/BDC.2014.15

The rest of the paper is organized as follows. Section II describes the related work, Section III describes the system architecture, Section IV describes the methodology, and GA problem formulation, Section V describes results with simulated data, and Section VI presents conclusions and future work.

II. Related Work

Previous works on scheduling in Data Grids [6][7]have been more concerned with the relationship between job assignment and data replication based on computation and data proximity. Mohammed et.al [8] discussed a Close-to-Files algorithm, searching the entire solution space for a combination of computational and storage resources for minimizing the processing time with the restriction of one dataset per job for execution.

Srikumar [9]described scheduling the distributed data intensive applications on global grids based on a set coverage approach for cost and time minimizing problems. This approach is based on the availability of both computation and data resources; however, data transfer from replicated sites and the selection of efficient computing nodes for minimizing the execution times are not addressed.

Big Data computing frameworks such as Apache Hadoop [10] is an open source implementation for MapReduce scheduling methods; the examples are Fair [11], Capacity [12], and Throughput [13]. Fair Scheduler is a pluggable group scheduler where in each group gets equal time slots for computation. Capacity Scheduler is similar to FIFO within each queue, but limiting the maximum resources per queue. Throughput Scheduler reduces overall job completion time on heterogeneous cluster by actively assigning tasks to computing nodes based on the server capabilities. Shared Scan Schedulers S^3[14]allows sharing the scan of a common file for multiple jobs arriving at different time intervals thus improving the performance of multiple jobs which are operating on a common data file.

In this paper, we discuss a scheduling methodology, where computational resources and data storages are decoupled with the data replicated over storage repositories which are geographical dispersed. Here, the problem is focussed on grouping the jobs based on the data requirements, and the objective is to minimize the total makespan considering both computational resources and communication bandwidth effectively.

III. System Architecture and Workflow

The system architecture is depicted in Figure 1 with four basic elements like scheduling broker, computing infrastructure providers, data providers, and analytics/applications developers/users. Compute providers offers a large scale computing infrastructure, data providers service the data on demand, scheduler broker periodically collects the jobs from the pool and determines the effective schedule to increase system throughput.

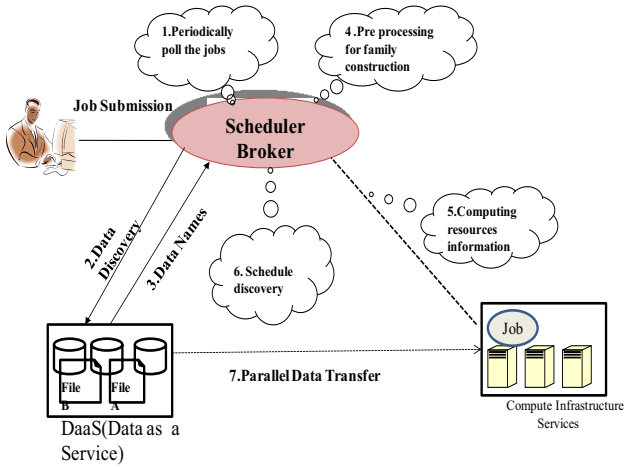


Figure 1. System Architecture

The workflow is depicted in Figure 2. The major activities in the workflow are: a) grouping the job based on the common/overlap data which we call as the family construction, b) determining the data workloads between compute and replicated sites, and c) discovering the optimal schedule to minimize the turnaround time of the jobs. These activities are explained below.

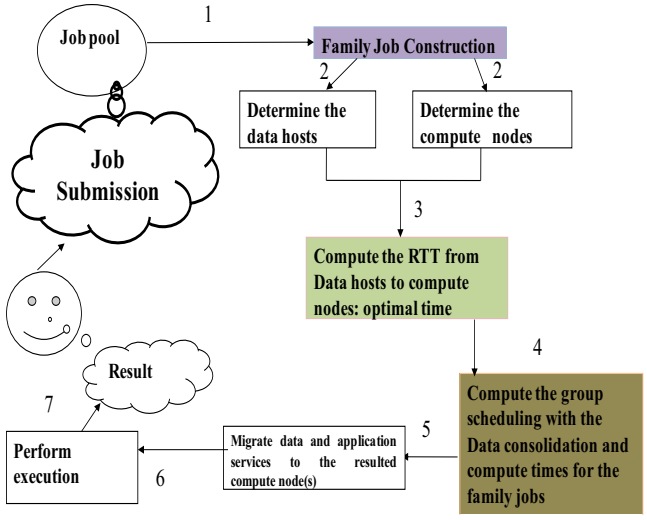


Figure 2. Workflow

i. <u>Family construction</u>: The jobs with common/overlap data are grouped together, called "**family**". To discover the grouping, metadata attributes such as object identifiers, and key/value descriptions are used as parameters. Object identifiers uniquely identify the

objects in a bucket, and the object metadata is a set of name-value pairs for describing the data content. Object based storage mechanisms such as Openstack Swift [15], and Amazon S3 [16] offers object keys and the associated metadata tagged with the files/objects. Object metadata is of two types- system metadata and user-defined metadata. System metadata describes the object creation dates, storage class information etc., and the user-defined metadata tags the additional information for the objects. First, we apply the query to discover the jobs with similar object identifier tags, followed by key/value pair combination for finding the common/overlap data. However, the discussed methods are limited, but, these could be extended to other data overlap/commonality computing techniques.

ii. **Determining data workloads:** The data workload is determined based on the available bandwidth between the replicated sites and the compute nodes. Network traces based on round trip time over a time period is used as parameter (weight factor for the data channel)in our model to estimate the amount of data to be transferred from each of the replicated sites.

iii. **Optimal Schedule:** The schedule is based on steady state genetic approach using turnaround time minimization as fitness value.

iv. **Data and applications migration:** Based on the schedule map, data and application services would be migrated to the compute nodes.

v. **Execution:** Jobs execution on the compute nodes, and the deletion of the temporary and migrated data sets from the computing nodes.

vi. **Result:** Final result sent to the end user.

IV. METHODOLGY

Here, we discuss the methodology for family construction, notations and problem formulation.

A. Family construction

Graph data structure, we call here as family graph is used for grouping the jobs. In the family graph, job is represented as node and the data required by both the jobs (adjacent nodes) is represented by the edge. A sample family graph with 7 jobs numbered from 1 to 7, and three data sets named from X_1 to X_3 are shown in Figure 3 .

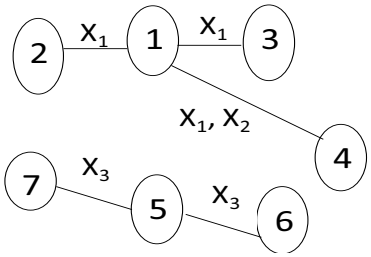


Figure 3. Family graph

The graph indicates that the jobs 1, 2, 3, and 4 require the data with id X_1, the jobs 1 and 4 require the data with id X_2, and jobs 5, 6 and 7 require the data with id X3 for processing. The families are formed by computing the connected components of the graph. The graph in Figure 3, results in two connected components with the nodes 1, 2, 3, and 4 for the first component, the nodes with 5, 6, and 7 for the second component. The resultant connected components form two groups or two family jobs which are to be processed further.

B. Notations used

Mathematical notations for the problem formulation are described in Table 1.

Table 1.Mathematical Notations

J = Total number of jobs
N= Total number of computing nodes in the grid
H= Total number of data service/providers
F= Total number of families
w_f= Total number of jobs in the family f.
TD_{fi} = Data Make Span(Consolidation time) in minutes of the family f∈F on Node i.
r_{hi} =Estimated packet transmission time in seconds between data provider h∈H to compute node i∈N.
X_f= Amount of data required in GB for the family f∈F.
x_{fhi} = Data chunk in GB from the data provider h∈H to compute node i∈N for the family f∈F.
ρ_{hi}= Weight assignment to the channel from data provider h∈H to compute node i∈N.
TR_{ji} = Turnaround time of the job j on node i.
TS_{ji}= Setup time of the job j on node i.
TA_j= Arrival Time of the job j.
Δ_{fi} = Decision variable.
δ_{fi} = Assignment variable.

C. Objective Function

The objective is to minimize the turnaround time of the jobs over the computing nodes.

$$\text{Minimize} \sum_{j=1}^{J}\sum_{i=1}^{N}\sum_{f=1}^{F} TR_{ji}\Delta_{fi}\delta^{f}_{j}$$

$$\delta^{f}_{j} = \begin{cases} 0 & \text{if } j \notin f \\ 1 & \text{if } j \in f \end{cases}$$

$$TR_{ji} = TD_{fi}/w_f + TS_{ji} + TL_{ji} - TA_j$$

where

- TR_{ji}: Turnaround time of the job j∈f on computing node i.
- TD_{fi}: Data consolidation of the family f on computing node i.
- TS_{ji}: Setup time of the job j∈f, on computing node i.

- TL_{ji}: Length of the job $j \in f$, on computing node i.
- TA_j: Arrival /Submission time of the job $j \in f$.

subject to the following constraints

- A family is assigned to either one compute node or none.

$$\sum_{i=1}^{N} \Delta_{fi} \leq 1 \text{ for } f = 1,2, \ldots . F$$

$$\Delta_{fi} \in \{0,1\}$$

- Compute node can have either none or many families assigned to it

$$\sum_{f=1}^{F} \Delta_{fi} \geq 0 \text{ for } f = 1,2, \ldots . N$$

$$\Delta_{fi} \in \{0,1\}$$

D. Determining the data consolidation time

Network traces between the computing resources and data hosts/providers are used to estimate the channel bandwidth availability. Such stored network traces over a time period for example 15 minutes are used as parameter to estimate the data quantity to migrate from each of the data providers to compute node. Below we will discuss the procedure for computing the percentage of the data to be obtained from each of the data providers to the compute nodes. Let us denote the compute node by *i*, data provider by h, job by j, and the family by f.

The families are constructed using the family graph as discussed in the section IVA. The family may have one or more jobs if they have common data. If each family consist of only one job, then F=J, otherwise F<J. Let w_f be the number of jobs in the family f also called weight of the family, then $w_1+w_2+\ldots..+w_F=J$. In the family job scheduling problem, data consolidation time is same for all the jobs that belong to the same family. Data Consolidation time is defined as the maximum time to consolidate the data from the identified data providers to the compute node. Data Consolidation time TD_{fi}, is the maximum time required to bring the data from the data provider(s) to the compute node *i* for the family f, and is defined as

$$TD_{fi} = \max_{h=1,H} (x^f_{hi} * r_{hi}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where x^f_{hi} is the chunk size and r_{hi} is the estimated time from the previous historical traces, for the family f from data provider h to compute node i. x^f_{hi} can be computed as below.

$$x^f_{hi} = \rho_{hi} * X_f \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Where X_f is total data size required for the family f, and ρ_{hi} denotes the weight assigned to the channel from host h to compute node i, is defined as

$$\rho_{hi} = (1/r_{hi}) / \sum_{i=1}^{H} (1/r_{li}) \ldots\ldots\ldots\ldots (3)$$

The time is estimated using the previous history of packet transmissions over a time period. Simplifying equation (1),using equations (2) and (3), we get

$$TD_{fi} = (1/r_{hi}) \Big/ \sum_{i=1}^{H} (1/r_{li}) * X_f * r_{hi}$$

$$= X_f / \sum_{i=1}^{H} (1/r_{li})$$

Family to node mapping is represented as the weight matrix. The problem can be solved as the bipartite assignment problem, but the limitations are, a node can have maximum of one family assigned, although it has enough processing elements for handling more than one family. Hence, this problem reduces to 0/1 knapsack which can be solved using greedy, dynamic programming or evolutionary techniques like genetic algorithms. In the proposed group scheduling, three possible schedules may occur for the computing nodes during execution, such as:

(i) No family assigned.
(ii) With exactly one family assigned.
(iii) More than one family assigned.

Based on the schedules described as above for a compute node, we discuss below the GA problem formulation, chromosome representation, scheduling algorithm and the results obtained with the simulated data.

E. GA problem formulation

Steady State GA [5]is used by replacing percentage of chromosomes across the generations until the solution converges or till the maximum number of generation is reached. Genetic algorithm library (GALib) [17] is used for implementation.

i. Chromosome representation

We choose integer based data structure for genomes representation. The gene index represents the familyid, and the genome indicates the compute nodeid to which the family is mapped. The data structure of the chromosome is shown Figure 4.

Gene	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
Gene Index	1	2	3	4	5	6	7	8	9

Node Id	2	4	3	2	4	5	3	2	1
Family id	1	2	3	4	5	6	7	8	9

Figure 4.Sample Chromosome

Consider F families to be mapped onto N computing nodes. Each family f will go to only one computing node, whereas a computing node may or may not be assigned with families. This yields a possible assignment size of N^F which is an exponential large value. Let us assume that there are 9 families and 10 computing nodes. Figure 4

represents that familyid=2 is assigned to nodeid=4, familyid=1 is assigned to nodeid=2, and so on. From the above figure it is also seen that nodeid=4 has been assigned with familyid=2 and 5; nodeid=2 has been assigned with familyid=1,4 and 8 and nodeid=6, 7, 8, 9 and 10 have not been assigned to any family.

ii. Evolve method

An initial population of chromosomes are selected at random and the fitness function is applied. The fittest chromosomes are carried to the next generation based on the Steady State overlap percentage of chromosomes. The process is repeated either till the maximum preliminary runs are completed or the convergence of the objective value is achieved. In order to choose the fittest chromosomes for the next generation the operators like crossover and mutations are used. The next generation chromosomes are created by genetically mating fitter individuals of the current generation.

iii. Scheduling algorithm

Table 2 describes the fitness function pseudo code and Table 3 discusses the proposed GA for scheduling the family jobs using the fitness function.

Table 2. Fitness Function pseudo code

Algorithm. Fitness function F pseudo code
Input: *Chromosome C*
Output: Turnaround time of the schedule
1 Turnaround time T:= 0
2 For all genes in the Chromosome C perform the following steps do
3 read gene index f and genome value i
4 Compute the jobs {J}∈ to f.
5 For all j∈J do
6 Estimate data consolidation time TD_{fi}, compute total jobs W_f in family f.
7 Compute setup time TS_{fi}, estimated job length on node i, TL_{ji} and arrival time TA_j.
8 Compute the turnaround time TR_{ji} of job j ∈J on computing node i.
9 $TR_{ji}= TD_{fi}/W_f + TS_{fi}+ TL_{ji}-TA_j$
10 $T := T + TR_{ji}$
11 end for
12 until end of chromosome
13 return T;

Table 3. GA for schedule discovery

Algorithm: Scheduling Algorithm pseudo code
Input: Population, Generations, Crossover percent, Mutation percent, Gene length, Percentage of chromosome to carry forward(P,g,c,m,l,r)
Output: A Schedule for all the jobs
1. Initial Run: Randomly generate population P chromosomes.
2. Repeat
3. Calculate the fitness of all chromosomes using **Fitness function F**
4. Arrange the population in the ascending order of fitness value
5. Copy the r best chromosomes to new population.
6. for the remaining chromosomes; perform the crossover with percent c and mutation with percent m. Copy the new off springs to new population.
7. Replace the current population with the new population
8. Until maximum generations or convergence.

V. Experiments and Results

We have used CloudSim toolkit [18] with its new capabilities for file replication, simulated object storage identifiers for the data sets to simulate the Big Data Clouds environment. We use a simulated network with computation and data storage nodes spread at several locations as shown in Table 4, depicting: (a) 4 locations CHYD, CBGL, CMUB, CDEL having 7, 6, 7 and 8 virtual computing resources. These 28 virtual compute resources provide an aggregate of 1400 processing elements.(b) 4 locations that provide 40 data storage nodes with corresponding simulated bandwidths.

The following experiments are conducted in the order described below.

- ***Data consolidation Analysis:*** Experiments are conducted to analyse the data transfer and consolidation timings from single site vs. the multiple replicated data sites considering the network traces over a time period.

- ***Determining optimal probabilistic values of genetic operators:*** Experiments are conducted to derive the optimal values of Genetic Algorithm (GA) operators for cross over, mutation and types of crossovers.

- ***Comparing with match making and heuristic techniques:*** The algorithm is compared with match making techniques like Data First, Compute First, and heuristics such as Simulated Annealing (SA).

- ***Comparison with Non family scheduling:*** The algorithm is compared with Non family i.e. without grouping the jobs.

Table 4. Simulated configuration of Big Data Clouds

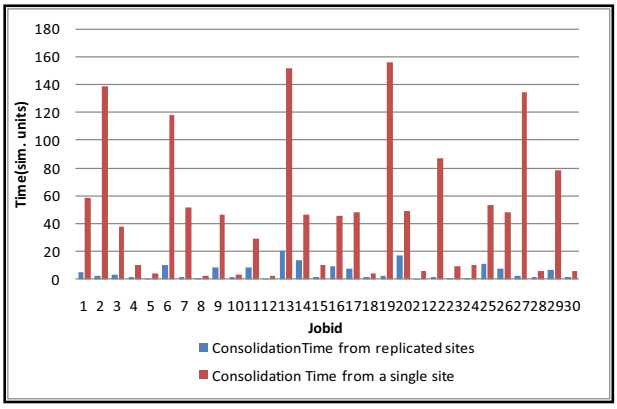
Resource Name	Type	Virtual processing elements/ Bandwidth
CHYD1-7*	Virtual Compute Provider	200/100Mbps-200Mbps#
CBGL1-6	-do-	500/50 Mbps-100Mbps
CMUB1-7	-do-	400/200Mbps-500Mbps
CDEL1-8	-do-	300/20Mbps-200Mbps
HYDS1-5	Virtual Data provider	0/100 Mbps to 200Mbps##
BGLS1-10	-do-	0/10 Mbps to 100 Mbps
MBS1-10	-do-	0/256 to 512 Mbps
RJPS1-15	-do-	0/56 to 128 Mbps

**1-7 indicates a total of 7 numbers.*
#200/100Mbps-200Mbps indicate randomly generated PEs with a maximum of 200 for each compute provider, with the network channel bandwidth randomly simulated between 100Mbps to 200 Mbps.
##0/100 Mbps to 200Mbps represents the randomly generated data source without computing elements, the bandwidth varying from 100 to 200 Mbps for a total of five data providers.

The experiment are simulated for a total of 1000 jobs with 40 virtual data providers and the network traces generated randomly between the virtual computing nodes and data providers. Based on the network traces average packet transmission times is estimated over a time period from data provider to computing nodes.

A. Data consolidation analysis

Graph 1depicts the data consolidation times when the single data storage and multiple replicated storage repositories are used. The results indicate that, data migration time from replicated sites is better than from a single site.



Graph 1. Data Transfer Times in replicated vs. Single

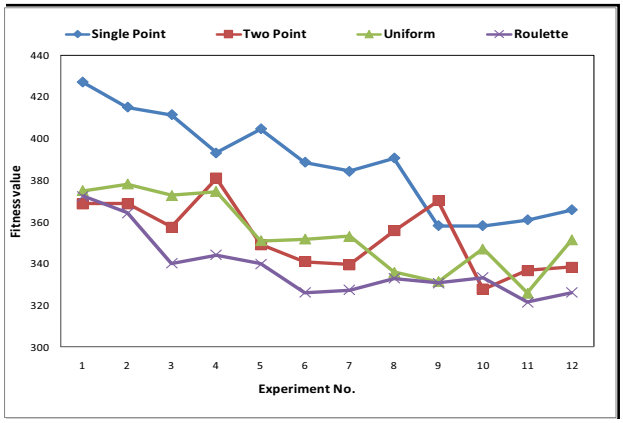
B. Determining the probability values for genetic operators

The experiments are conducted to determine the genetic operators and the probability values essential for the GA operators, for an upper limit of 1000 jobs over a schedule period is discussed. Several experiments are conducted to determine the crossover operator among one point, two point, uniform, and roulette wheel. The experiments are performed by fixing the cross over and mutation operators to 0.9 and 0.01 and varying the population length and generations to study the convergence of the fitness value. The conducted experiments are shown in Table 5, the resultant fitness values from the experiments are depicted in
Graph 2.

Table 5.Experiments to determine genetic operators

Exp. no	Pop. Length	Total no. of generations
1,2	100	100
3,4	100	200
5,6,7	200	100
8	200	100
9,10,11,12	200	200

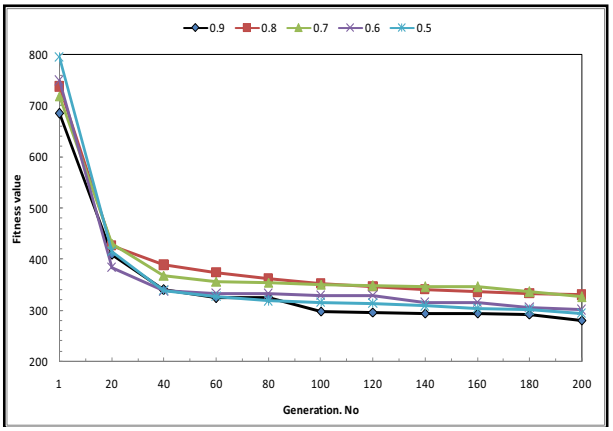
The experiments indicated in
Graph 2 determine the roulette wheel has better convergence across the generations while compared to the other genetic operators. Hence, the roulette wheel operator is selected for cross over operations for the next level experiments.



Graph 2. Fitness value comparison for genetic operators

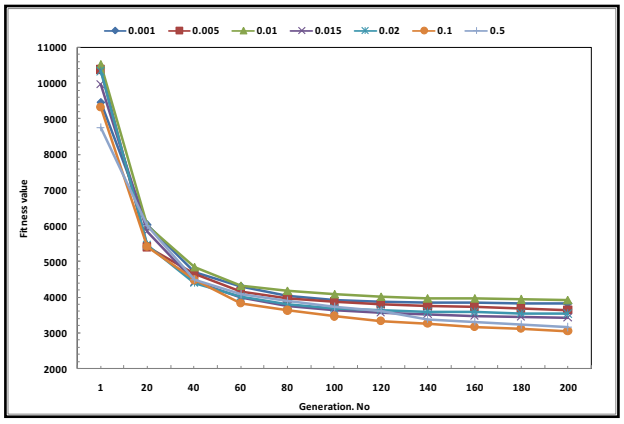
In the next step, the experiments are conducted for determining the cross over and mutation probabilities while fixing the roulette wheel cross over operator. The experiments are conducted for 200 generations, with an initial cross over probability value of 0.5, up to 0.9, with an increasing value of 0.1 in each step. The corresponding fitness values obtained are depicted in

Graph 3. The results indicate roulette wheel cross over operator with the probability value of 0.9 has better convergence over the other experimented probability values.



Graph 3. Fitness value convergence across generations

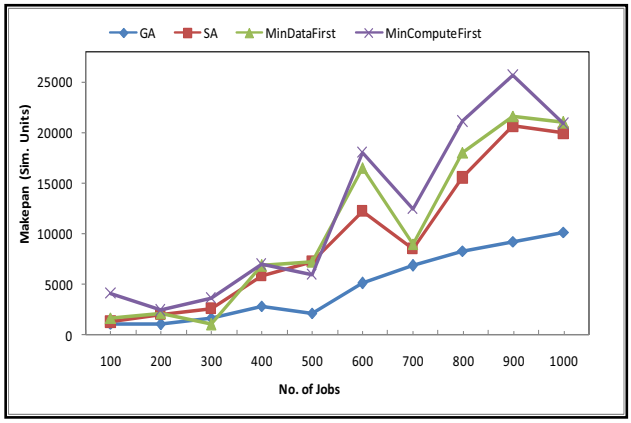
Experiments are performed with varying mutation probabilities to determine the appropriate mutation ratio by fixing cross over operator of 0.9 and roulette wheel, the experiments shown in Graph 4 indicates with mutation probability of 0.1 has the better convergence fitness value.



Graph 4. Mutation probability

C. Comparing with other techniques

The proposed GA is compared with match making and heuristic techniques discussed below. Here, two types of match making techniques such as Minimum Data consolidation First and Minimum Compute First are used. Later, heuristic technique such as Simulated Annealing techniques is discussed. The obtained results are compared with the proposed GA approach which is depicted in Graph 5.



Graph 5. Comparing GA with other techniques

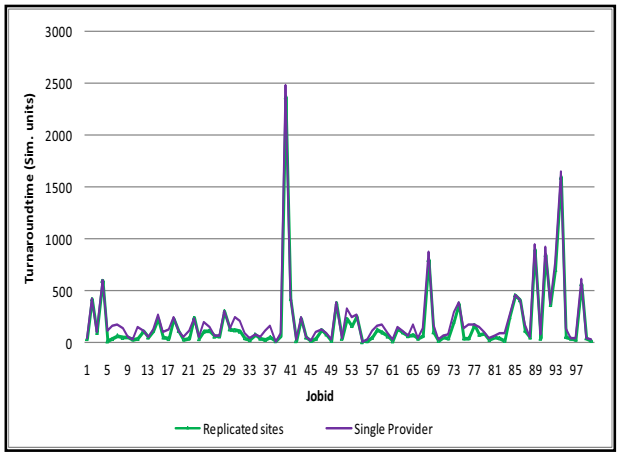
- ***Minimum Data consolidation First at node*** – In this mapping, a compute resource that ensures minimum data consolidation time is selected for the family.
- ***Minimum Compute First*** – In this mapping, a compute resource that ensures the minimum computation time is selected for the family.
- ***Simulated Annealing-*** In this mapping heuristics are applied by discarding the worst fit values from the current state to the next state and moving towards the best selection.

The results indicate that Minimum Compute First technique has resulted in larger makespan while compared to Minimum Data First and SA techniques. Minimum Data First and Simulated Annealing techniques have almost the same makespan value with performance better than Minimum Computer Fist technique. However, the proposed GA has resulted in minimal makespan while compared to matchmaking techniques and SA. This could be due to the natural evolution procedures of GA and fitness functions used to obtain the near optimal solution.

D. Performance comparison of family vs. non family scheduling

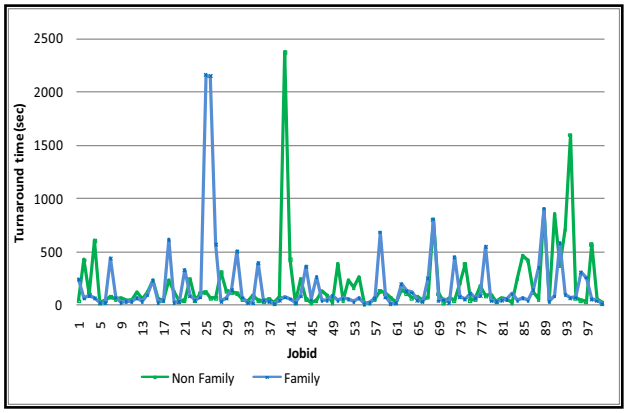
Experiments are conducted for analyzing the turnaround times for the families and non-families from a single storage vs. replicated storage. The results depicted in Graph 6, illustrate that turnaround time is less from the replicated sites while compared to the results from the single site, which is due to the minimal data consolidations from replicated sites.

Another set of experiments are performed for analysing the turnaround times of family vs. non family scheduling. This is carried out by applying the data migration/consolidation from the replicated sites to the selected computing nodes.



Graph 6.Turnaround times of Non Family

Graph **7** illustrates, the jobs with the family grouping has resulted in minimal turnaround time while compared to the non-family. This is due to the data consolidation carried out one time for the entire family job. This would reduce the data migration overheads for each job and reduce the network bandwidth consumptions. However, for few jobs the resultant turnaround time is more while compared to non-family scheduling, which could be due to the grouping that has resulted in longer data consolidations and computing times for the jobs. The longer data consolidations is due to more numbers of jobs in the family, and the longer computing times is due to the availability of minimal computing elements at the selected compute node than actually demanded for processing.



Graph 7. Turnaround Times for Family vs. Non Family

VI. Conclusions and Future Work

The proposed family/group scheduling model addresses the data intensive problems to minimize the turnaround time of the jobs where the computing and data resources are decoupled. The jobs with common data are grouped together, based on the family graph and connected components to which a parallel data approach is applied.

Steady state GA is applied to discover the optimal schedule. The results are illustrated for the both family and non-family schedules from a single site and multiple replicated sites. The results indicate that, data migration from replicated sites show performance improvement over a single site. The experiments also show that family schedule performs better over the non-family schedule, whenever the grouped jobs do not exceed the available node capacity.

The connected components of the graph are used for grouping, which is a compute intensive process. In future, it is proposed to use Rough Set theory for the grouping. The algorithm is tested with time shared scheduling policy. In future the studies would be conducted on space mechanisms such as buddy system, DHC (Distributed Hierarchical Control), Ouster out matrix, and bin packing. The algorithm would be modified to map the family job to the node where data is already present, which would eliminate the data consolidation time.
The system throughput is decreased while the family capacity exceeds the available node capacities. Hence, a study is required to schedule such families, by adding a penalty to the total compute time, so that the better node could be selected for scheduling. This paper addresses the migration of the data based on network traces over a time period; however, a detailed study is required to train the system for different network traffic conditions. The proposed algorithm can be extended for deadline and budget constraints. Presently, the model executes the jobs after the data is consolidated for the family; however, the studies can be conducted for the execution soon after the data for the job is made available.

ACKNOWLEDGMENT

We express our thanks to Prof. Hrushikesh Mohanty, and Prof. Rajeev Wankar from University of Hyderabad, India for providing the valuable guidance. We express thanks to Dr. Sudheer Reddy of ADRIN for discussion on GA. We thank Smt. GeetaVaradan, Director ADRIN for her support and encouragement in pursuing the research.

REFERENCES

[1] M. D. Assuncao, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, Big Data Computing and Clouds: Trends and Future Directions, Journal of Parallel and Distributed Computing, Available online 27 August 2014, DOI: 10.1016/j.jpdc.2014.08.003

[2] Apache MapReduce, http://hadoop.apache.org/docs/stable/mapred_tutorial.html(11.06.2014).

[3] A. Chaudhri, R. Kune, K.P. Kumar, and G.Varadan, High Performance Private Cloud for Satellite Data Processing– Engineering in Cloud, International Conference on Advances in Cloud Computing , ACC 2012 , Bangalore , India.
[4] Z. Michalewicz: Genetic Algorithms + Data Structures =Evolution Programs. 1992, Springer.
[5] Goldberg D.E., Genetic Algorithms in search, Optimization and Machine Learning, Addison Wesley , Reading , MA,1989.
[6] K. Ranganathan, and I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications", Proc. 11th IEEE Symposium on High Performance Distributed Computing (HPDC). Edinburgh, UK, USA, July 2002.
[7] T. Phan, K. Ranganathan, and R.Sion, "Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm", Proc. 11th Workshop on Job scheduling Strategies for Parallel Processing. Cambridge MA: Springer-Verlag, Berlin, Germany, June 2005.
[8] H. Mohamed, and D. Epema, "An evaluation of the close-to-files processor and data co-allocation policy in multi-clusters", in Proc. 2004 IEEE International Conference on Cluster Computing, San Diego, CA, USA, Sept. 2004.
[9] S. Venugopal, Scheduling Distributed Applications on Global Grids, Ph.D. Thesis, University of Melbourne, Australia, July 2006.
[10] Apache Hadoop, http://hadoop.apache.org/ (15.06.2014).
[11] Fair Scheduler , http://hadoop.apache.org/docs/r1.2.1/fair_scheduler.pdf(11.06.2014)
[12] Capacity Scheduler, http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.pdf (11.06.2014)
[13] S. Gupta, C. Fritz, R. Price, J. D. Kleer, and C. Witteveen, Throughput Scheduler: learning to schedule on heterogeneous Hadoop clusters, Proceedings of the International Conference on Autonomic Computing, ICAC 2013, June, 2013, San Jose, CA, USA.
[14] L. Shi, X. Li , and K.L. Tan, S3: An efficient Shared Scan Scheduler On MapReduce Framework, International Conference on Parallel Processing, ICPP 2011, Taipei, Taiwan, September 2011.
[15] OpenStack Swift, Object Based Storage and REST Services, http://swiftstack.com/openstack-swift (5.3.2014).
[16] Amazon S3: www.aws.amazon.com/s3(2.2.2014).
[17] Java GA Lib, Genetic Algorithm Library: http://sourceforge.net/projects/java-galib (11.06.2014).
[18] R. Calheiros, R. Ranjan, A. Beloglazov, C. Rose, and R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience, 41(1): 23-50, Wiley Press, New York, USA, January 2011.