Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

# A Review on Parallel Genetic Algorithm Models for Map Reduce in Big Data

Kiranjit Pattnaik
School of Computer Engineering
KIIT University
Bhubaneswar-751024, Odisha, India.

Bhabani Shankar Prasad Mishra
School of Computer Engineering
KIIT University
Bhubaneswar-751024, Odisha, India.

***Abstract* - Map Reduce is the basic software framework used in the field of Big Data because of its high scalability. The use of parallel genetic algorithms in map reduce gives more accurate close to optimal value and due to its parallel nature huge volume of data can be handled properly. This survey aims to collect and organize most of the recent publications on the parallel genetic algorithms using map reduce. This paper presents a comparative view of different parallel models of genetic algorithm for map reduce and also provides a detail information on different platform and tools for map reduce.**

***Keywords - Parallel Genetic Algorithm, Map Reduce, Big Data, Map Reduce using Global Parallelization, Map Reduce using Coarse Grained Parallelization, Map Reduce using Fine Grained Parallelization.***

## 1. INTRODUCTION

Big data is a new generation of architectures & technologies, which extracts data from very huge volume of different varieties of data and by implementing high velocity capture, discovery and/or analysis.
Basically there exist 3 V's of Big Data ([1], [2]) as

- Volume - How much data (Yottabytes, Pettabytes…)
- Velocity - How fast the data is processed
- Variety - Different types of data

But advance research has shown that only these 3 V's of Big Data is not sufficient enough to describe the Big Data. So, 2 more V's of Big Data has been summarized.

- Veracity - Uncertainty of data (Data incompleteness)
- Value - Turn Big Data into values else useless (business perspective)

Big data is a combination of the new and old data-management technologies that has been developed over time. To gain the true insights in the organizations Big Data allows organizations to manage, manipulate and store the huge volume of data at the required time and precise speed.

Map Reduce [3] deals with a large volume of data. Evolutionary algorithms can be used in Map Reduce for further optimization of the data. EAs like Genetic Algorithm can be applied to large search space due to its efficient use of operation like selection, crossover and mutation. As GA operates one time consuming so, parallelization of GA can be a solution to it. The parallel GA does parallel search from multiple points in the space and has higher efficiency and efficacy than the sequential GAs. It works on the coding of the problem and gives a better search even if no parallel hardware is used. The parallel GAs yield alternate solutions to a problem and is basically robust in nature and has access to easy parallelization as islands or neighborhood as well as has an easy cooperation with other search procedures.

This paper explains how different PGA models can be combined in Map Reduce ([4], [5]) technique for the abstraction of huge data set. The rest of the paper is organized as below. Section 2 describes about the Map Reduce technique. Section 3 describes about different PGA models. PGA based Map Reduce is described in Section 4. Comparison of different PGA models and conclusion is described in Section 5 and 6 respectively.

## 2. MAP REDUCE

Map Reduce is a framework in which huge data sets are processed parallely using a large number of computers or also termed as nodes. The nodes are denoted as a cluster or grid depending if the nodes are all on the same local network using similar hardware or are shared across distributed network using heterogeneous hardware. The data is stored either in an unstructured file system or in a structured database.

Map Reduce is called the heart of Hadoop. Huge scalability is provided across hundreds or thousands of servers in a Hadoop cluster. The Hadoop programs perform two distinct tasks of Map Reduce. The first being the Map task, where a set of data is taken as input and then changed into another set of data for further processing, where each individual component is reduced into key/value pairs. The second is the Reduce task whose input is the output produced by the Map task and then it combines those key/value pairs into another set of key/value pairs. The reduce task always comes into action after the Map task. The basic phases involved in map reduce are:

- Map: The map function is applied by each worker to the local data, and writes the output to a temporary storage. Here the master node see's that if duplicate copies of the input data is present then only one of them is processed.
- Shuffle: The data produced by the map function is distributed across the worker nodes based on the output keys such that data belonging to a particular key is located on the same worker node.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

- Reduce: The reduce function is applied on each worker node to process the output data from the shuffle phase as per key/value in parallel and produce the final output.

Map Function : Map (key 1, value 1) -> list (key 2, value 2)

Reduce Function : Reduce (key 2, list (value 2)) -> list (key 3, value 3)

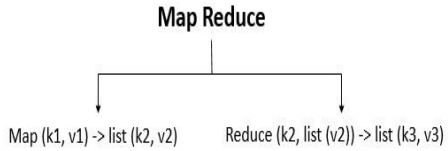


Figure 1: Two Phases of Map Reduce

### *2.1 Tools and Environment*

Due to massive parallelization and scalability of Map Reduce it's being used by many tools such as the Apache Hadoop, CouchDB, Disco Project by Nokia Research, Infinispan (a successor of JBoss Cache), MangoDB and Riak. The environments in which Map Reduce is basically used are Windows 7/8, Linux (Ubuntu), Virtual Machine, Cloud System.

### *2.2 Basic Steps for Map Reduce*

1. Preparing the Map function for the input in which Map processors are assigned to the input key value k1 on which each processor would work and all the input data are provided to that processor associated with the key value.
2. Executing the Map function i.e. the for each key value k1 the map function is executed once which would generate output organized by new key values k2.
3. Shuffle and Sort the Map output to the designated Reduce processors, the key value k2 is assigned to each processor and Map generated data is provided to that processor with that key value.
4. Execute the Reduce function once for each key value k2 produced by the Map function.
5. Final Output is produced by the Map Reduce system and all the Reduce function output are collected and sorted as per key value k2 to produce the final result.

### *2.3 Pseudo Code*

1. Divide the input into 'm' parts where 'm' is number of mappers and send it to each mapper.
2. Execute the map function on the input for each mapper.
3. Combine the similar sets key/value pairs and send it to the shuffle/sort phase.
4. Sort the intermediate key/value pairs as per the key.
5. Execute each list of intermediate key/value pairs by the reduce function (if there exists 'r' reduce phases then divide it into 'r' part and execute the reduce function)
6. Summarize or combine the output of the reduce function to form the output

## 3. PARALLEL GENETIC ALGORITHM (PGA)

Genetic algorithms [6] are stochastic search algorithms centered on ethics of natural selection and recombination. It helps to find the optimal solution to the problem by handling the population of the required solutions. The evaluation of the population is done and the best solutions are selected out of it to mate & reproduce to become candidates for the next generation. After a series of generations, the good traits lead the population, which results in the better quality of the solutions.

The Darwinian evolution is the basic mechanism of the genetic algorithms in which the bad traits are excluded from the population as they appear in individuals which cannot last the process of selection. On the other hand the good traits survive and are mated with others to form better individuals. There also exists an operator in GA termed Mutation which is considered a secondary whose purpose is to guarantee that diversity in the population is not lost, so that the GA can carry on its search. Crossover and Mutation are the two operators in simple GAs which are centered on the natural genetics to explore the search space. The genetic algorithm unlike other optimization techniques apply various genetic operators on the population and are modified for the next generation until a global optimum is found out. For nontrivial problems high computational resource i.e. larger memory and more number of search is required. With this goal many new advances are achieved continuously by designing or using new operators, termination criteria, different hybrid algorithms, etc.

Parallel Genetic Algorithms (PGAs) are not only just the parallel versions of the sequential genetic algorithms but having a parallel algorithm whose behavioral results is better than the sum of the individual behaviors of its constituent sub-algorithms.

### *3.1 Types of PGA Models*

There are different types of parallel genetic algorithm (PGA) models and the parallelization is done on the population i.e. some use single population while others use each individuals or the population being sub divided into various subpopulations. Of these some methods can achieve massive parallel computer architectures while others to multicomputer with some powerful processing elements. Basically there exist three models of parallel genetic algorithm ([7], [8], [9]) and they are

1. Master Slave Model GA (Global Parallelization)
2. Grid Model GA (Fine Grained Parallelization)
3. Island Model GA (Coarse Grained Parallelization)

In a Master Slave GA or the Global Parallelization there exists a single population, but the fitness evaluation is scattered among several different processors. In this model, the whole population is considered by the selection and crossover operators. In the Global Parallelization the population is managed by the master process and the individuals are forwarded to the slave processors for evaluation. After the evaluation is complete, the results are collected by the master and the genetic operators are

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

applied to yield the next generation. This architecture can be implemented on both shared memory multiprocessors as well as distributed memory machines including networked computers. The architecture of the Master Slave Model is shown in Figure 2 :

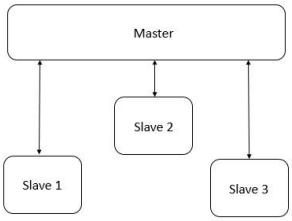


Figure 2: Architecture of the Master Slave Model

In the Grid model or the Fine Grained parallelization all the operations are done on each individual rather than on the whole population. This model consist of one spatially structured population and is suitable for massive parallel computers. In this model selection and mating are limited to a small neighborhood, but the neighborhoods overlap allowing some communication among all the other individuals. The ideal case would be having only one individual for each available processing element. The architecture of the Grid Model is shown in Figure 3 :

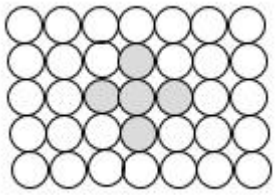

Figure 3: Architecture of the Grid Model

The Coarse Grained parallel GAs are also very much similar to the master slave model in structure but it mainly works with the sub-populations i.e. the population is sub divided into various groups and each group is then mapped separately and executed and it follows a new method termed migration which works with each individual chromosomes and the best one is migrated to the next step. In this model on each subpopulation the GA operations are performed and the subpopulations interchange data information by permitting individuals to allocate from one island to another as per the given criteria. The architecture if the Island Model is shown in the Figure 4 :

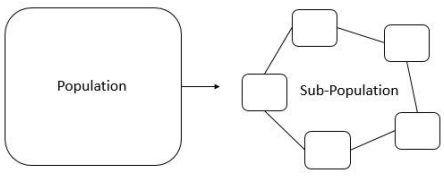


Figure 4: Architecture of the Island Model

## 4. PGA BASED MAP REDUCE

There are three basic models of PGA like Global Parallelization, Coarse Grained Parallelization and the Fine Grained Parallelization. In the Map Reduce framework the map and the reduce phase is user defined and it can be modified as per the requirements. In such a scenario the Parallel Genetic Algorithms is used in its map and the reduce phase accordingly following the specific parallelization method.

### *4.1 Map Reduce using Global Parallelization*

The Global Parallelization follows the Mater-Slave model in which a computing node becomes the master and the remaining computing nodes are the slaves. Most of the genetic algorithm operations on the population is performed and stored in the master node. The master sends one or more chromosomes to the slaves and then it waits for the results returned by the slaves. Similarly the work of the slaves is to accept the population and perform the selection and required genetic algorithm operations and send the selected best individual to the master node. A common architecture of Map Reduce using Global Parallelization is shown in Figure 5 :

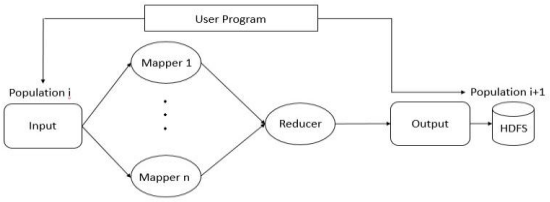


Figure 5: Architecture of Map Reduce using Global Parallelization

The main program is responsible to create an initial random population and to start the evolutionary process by invoking the Input Format. Moreover, it is also responsible to stop the process according to some termination criteria, such as population convergence or a max number of performed executions.

Input : The Input module is in charge to read the source code from the data store and to assign the chromosomes of the current generation to a bunch of Mappers. To parallelize the computation of the fitness function for each chromosome, each of them is assigned to a different Mapper. This is achieved by generating a unique key for each chromosome, which will be calculated as a function of the chromosome, current generation, and specific unit as per the requirements. It is worth noting that unit as per requirements is considered and in this way the model supports the application of multiple instances of GA to different units.

Map : The Mapper is responsible to evaluate each received chromosome exercising the unit with the corresponding test data and observing the software behavior to compute chromosome fitness value such as branch coverage as an example. Then, the Mapper generates an intermediate pair of key and value, where the key is needed to properly assign the Reducers, while value is the pair of the chromosome and the fitness value. By assigning the same key to all intermediate pairs we can realize a global parallelization model which forces the model to use exactly one Reducer.

Reduce : The Reducer module is invoked only once when all the Mappers terminate their execution. Once all the pairs are connected, it applies the selection on the entire population and then the evolution operators is applied so as to obtain a new offspring.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

Output : At the end of each iteration, the Output saves the data related to the new offspring into the data store. The Master module manages the overall computation together with the resource assignment. As the Input begins to emit the key and the value pairs, the Master module assigns the chromosomes to the Mapper. Similarly, when Mappers emit he intermediate pair of kay and value the Master module is responsible to collect them to properly assign the input to the Reducer module. Finally, once the Reducer outputs the new offspring, the Master module notifies the User Program to check according to the stopping criterion if the computation should be terminated or restarted by invoking the Input on the new offspring.

Dino Keco et. al. [10] proposed 'Parallelization of Genetic Algorithm using Hadoop Map/Reduce' in 2012 in which he performed a modification on the paper named 'Scaling Genetic Algorithm using Map Reduce' by A. Verma et. al. [11]. Verma et. al. used only one map reduce phase for each generation of genetic algorithm, and new map reduce phase is executed for each generation, while for all generation of the genetic algorithm Dino Keco et.al. used only one map reduce phase. The algorithm proposed by Dino et. al. is shown below :

*Algorithm 1 :*
Input Phase (s : Size of Population, nom : Number of Maps)

s = populationsize();
nom = Numberofmaps();

*Algorithm 2 :*

Map Phase (c : CurrentGeneration, p : Population, s : PopulationSize, l : LastGeneration, f : FitnessFunction, ng : NextGeneration, o : Optimal, os : Offspring, pa : Parents)

c = 0;
p = GenerateRandomPopulation(s);
f(p);
while (c < l)
p=ng(p);
f(p);
c++;
end while;
o = best in f(p);
return o;
ng : for (unit : p)
pa = selection(p);
os = crossover(pa);
os = mutation(os);
Add os to new Offspring

*Algorithm 3 :*
Reduce Phase (o : Optimal, co = CheckOptimal)
For (unit : o)
co(unit)

Federica Sarro et.al. (2012) [12] proposed a 'Parallel Genetic Algorithm based on Hadoop Map Reduce for the automatic generation of Junit Test suites' using the Global Parallelization in which the solution is created on Hadoop Map Reduce and since it supports cloud as a work environment and on graphic cards thus high scalable parallelization of GA is achieved. His preliminary analysis was to calculate the speed up with respect to the sequential execution. His idea in using Map Reduce is to parallelize the Genetic Algorithm along with the evaluation of the chromosome fitness and even encapsulating each iteration of the genetic algorithm as a distinct Map Reduce job allocating it to several other mappers, while to collect the results and to perform the various genetic operations such as selection, crossover and mutation to produce a new generation only a single reducer is responsible following the master slave model or global parallelization. The algorithm presented in the paper for the parallel genetic algorithm is mentioned below :

*Algorithm :* (SUT : SoftwareUnderTest, P : Population, mpj : MapReduceJob, po : PostExecutionFunction, pr : PreExecutionFunction, ic : InstrumentedCode, sc : StopCriteria)

P = pr(SUT);
while (!sc)
mpj = create mjp;
P = mpj.execute(P);
End while;
po;
pr(SUT) : P
ic = instrument(SUT);
move(ic, HDFS);
P = CreateRandomPopulation;
Return P;

The module of the Parallel Genetic Algorithm lets the user to manage the execution of GA and specify the SUT (Software Under Test). Upon termination of GA, for the software under test a test suite is returned to the user. To get material about the analysis in a program run a pre execution phase is required to instrument the SUT bytecode as well as to generate a random initial population. Then, by creating and executing the map reduce job the evolutionary process is started and at each iteration the termination criteria is checked. To clean the local and the distributed file system a post execution phase is needed.

The proposed algorithm is carried out in 3 steps:

Split - In this phase the input acquires the current population from the HDFS and is divided to be distributed among the mapper modules. On the availability of number of mappers the input split is computer dynamically. The master module assigns the resources and the computations and maintaining the load balance. When the input begin to emit the key and value pair, then the master component becomes active and assigns the spilt to different available mappers.

Map – In this phase each task is carried out by each mapper parallely. For evaluating the corresponding fitness value a class of SUT is implemented with the test suites related with the chromosomes for observing the software behavior. Upon the completion of the evaluation a new key value

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

pair is generated by each mapper where value is a pair of fitness value and chromosome while the key would be used by the master to assign the task to the reducers. The generated key for all chromosomes would be the same in order to have a single reducer. The master module collects the output produced by the mappers which would become the input for reducer.

Reduce - Once the output of the mappers are collected by the master the reducer receives an entire population where fitness value of all the chromosomes have already been assigned. Thus, survival selection, crossover and mutation operators are used on the new generation by the reducer to produce a new offspring which would be calculated in the next map reduce job. Using the record writer the new offspring is saved into the HDFS by the output format, allowing the Parallel Genetic Algorithm to validate the stopping criteria. The work of the master module in this phase is to restart the computation of the PGA invoking the map reduce job for the new offspring created by the reducer.

### *4.2 Map Reduce using Coarse Grained Parallelization*

The Coarse Grained parallelization follows the Island model in which different computing nodes are set commonly termed as islands in this case. In this model the population is further sub divided among the nodes and the genetic algorithm operations is executed on each computing node on its own sub-population. For the conformity of good solutions an operation termed Migration is carried out in which exchange of the chromosomes is done among the nodes with a certain probability in which a node sends the chromosome to the other nodes and gradually the other nodes replace a chromosome and then a migration strategy is used in which the node which replaced the chromosome in their population should be chosen to migrate or replace.

Some modules of the above architecture remain unchanged (i.e., User Program, Reducer, Output Format, and Master), while others have to be modified, i.e., Input Format and Mapper. In particular, we have to define a partition strategy able to assign each island of chromosomes to a different Reducer. This can be obtained by modifying the algorithm necessary to generate the keys in the Input Format. Now the key is calculated considering also the island that a chromosome belongs to. As for the Mapper, it has to emit a pair <key, value> where value is computed as in the previously described architecture, while key is now function also of the island containing the chromosome. As a result, each chromosome belonging to the same island, in the same generation and exercising the same code, will be fed to the same Reducer. Also in this architecture, the Reducers are responsible to apply selection and evolution operators for the new offspring, making it available to the Output Format. Finally, since the coarse-grained model strongly relies on the concept of migrating chromosomes among islands, this duty will be in charge of the Mapper module that randomly changes the island used in the computation of the key value, with the consequence that the involved chromosomes will be assigned to a different Reducer. The overview architecture is shown is the Figure 6 :

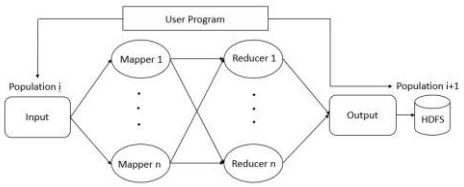


Figure 6: Architecture of Map Reduce using Coarse Grained Parallelization

An extension of the coarse grained model proposed by Chao Jin et. al. (2008) [13] named 'MRPGA : An Extension of Map Reduce for Parallelizing Genetic Algorithms' in which there exists another Reduce phase including the Map Reduce phases i.e. 2 Reduce phases. The architecture of the proposed model is mentioned in the Figure 7 :

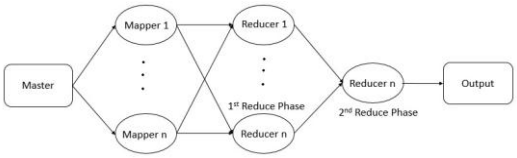


Figure 7: Architecture of an Extension of Map Reduce using Coarse Grained Parallelization

*Algorithm 1 :*
Map Phase (m : MapperFunction, I : IndividualValue, d : DefaultValue)

M(key, value);
P = A1[0], A2[0], … , An[0] = I ;
P1 = Evaluation (A1[0], A2[0], … , An[0]);
Emit (d, P1);

*Algorithm 2 :*
1st Reduce Phase (r : ReducerFunction, I : IndividualValue, k : Key, v : ValueList)

R(k, v);
i=0;
for (each value in v)
P[i] = A1[i], A2[i], … , An[i] = I ;
i++ ;
end for;
P1 = Selection(P);
For (each value in P1)
Emit (I, 1);
End for;

*Algorithm 3 :*
2nd Reduce Phase (k : Key, v : Value, I : IndividualKey, f : FinalReduceFunction)

f(k, v);
P = A1[0], A2[0], … , An[0] = I ;
Emit(P, 1);

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

*Algorithm 4 :*
Coordinator(sts : SendToSchedulerFunction, rfs : ReceiveFromSchedulerFunction)

MapReduceGA();
t=0;
P[0] = A1[0], A2[0], … , An[0];
Evalute (A1[0], A2[0], … , An[0])
While (!T(P[t]])
P1[t] = Crossover(P[t]);
P1[t] = Mutation(P1[t]);
Sts(P1[t]);
P[t+1] = rfs(t);
T = t+1;
End while;
Return P[t];

### *4.3 Map Reduce using Fine Grained Parallelization*

Fine Grained Parallelization is known to be the highest level of parallelism. It follows the Grid model in which the computing nodes are arranged in a grid type or a spatial structure and each node having a single chromosome where communication of the nodes is done with the neighboring nodes. Here the collection of all the chromosomes in each node is termed as the population and due to the overlapping of the neighborhood the good traits of a superior individual eventually can extent to entire population.

As final level of parallelization, a fine-grained model can be achieved with a slight variation in the last proposal. In this model, each chromosome is assigned to a single Mapper and then to a randomly selected Reducer. This can be obtained by acting on the way the Mapper module produces key values, which now are pseudo-randomly generated. (Pseudo-Random function is a way to implement the neighboring required by the fine-grained model.)

Rohit Konkedar et. al. (2012) [14] proposed 'A MapReduce based Hybrid Genetic Algorithm using Island Approach for solving Time Dependent Vehicle Routing Problem' in which a distributed Island model PGA is implemented where the execution of its own algorithm and the maintenance of the sub population is carried out by each node where probabilistic selection and replacement based on fitness is used. Crossover and mutation is performed on each island thereby building a diverse population. The proposed model architecture is shown in Figure 8 :

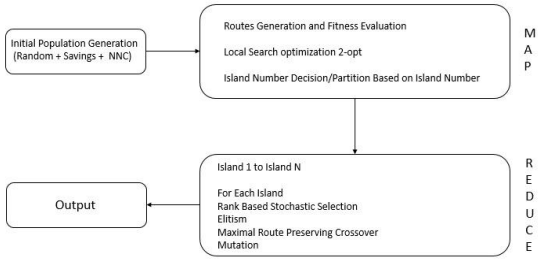


Figure 8: Architecture of the Map Reduce using Hybrid Parallelization

Steps involved in the proposed algorithm :

1. Generate the initial population using Random search and the Savings Algorithm in which the route is constructed and listed in decreasing order as well as by using the NNC Algorithm which is an improved algorithm of the nearest neighbor algorithm for the constructing the route. The population size is maintained to be around 200 per reducer.
2. Map Phase : In this phase fitness evaluation and 2-opt local optimization is carried out. The 2-opt local optimization is a local search algorithm in which the neighborhood exploration is enhanced by maintaining a certain probability by removing the two edges from the route and connecting the two paths created. Island number is assigned randomly to each individual with results as key and value pairs, in which key is the island number and value is the mixture of string representation of individual and the fitness in which the routes are separated by '#'.In the following map phases key is the island number of the previous phase and same operations are performed on the individuals.
3. Reduce Phase : The individual would go to that reducer partition which is decided by the island number. In the reducer, the individuals are ordered in ascending order as per the fitness value and two individuals are chosen for the crossover operation by using the ranking method. Maximal routes preservation and mutation is done by the Maximal route preserving crossover (MRPC). With certain probability migration is carried out by changing the key which is the island number of each individual. Individuals are produced in form of key and value pair, where the island number is termed the key and the child generated from crossover would be the value. The individual having maximum fitness during the selection process for elitism is emitted. The maximum fitness of each reducer is written in separate files, which are grouped after each map reduce phases and then are sorted and read for testing the termination norms. Until the termination norm is satisfied these steps are repeated.

## 5. DISCUSSION

Various different parallel genetic algorithm models using map reduce has been stated before and now a comparison is made between these models along with the environments used.

### *5.1 Comparison*

Verma et. al. (2009) provided another approach for using MapReduce for parallel GAs and highlighted several concerns and limitations of the proposal described in Gin et. al. (2008). The main concern was about scalability of parallel GAs. The previous proposal max limit was 32 nodes, due to the essential serial organization imposed by the use of a single coordinator, which performs mutation, crossover, and evaluation of the convergence criteria. Furthermore, they highlighted that the extension proposed in Gin et. al. (2008) could easily be implemented in the traditional MapReduce model using a Combiner instead of the proposed local reduce, as shown in Dean et. al. (2008). Finally they pointed out that using the same value as keys

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

produced by the mapper, local reducer and the reducer, MRPGA does not hire any characteristics of the Map Reduce model (i.e., the shuffle and the group by keys). Differently, Verma et al. (2009) exploited the traditional MapReduce model by implementing GA in Hadoop to solve the onemax problem. They investigated scalability and convergence of the proposed GA and found that larger problems could be resolved by adding more computational resources without changing algorithm implementation.

The model presented by A. Verma et. al. (2009) had a big input output imprint because full population is saved to HDFS after each generation of GA and due to this huge performance degradation is caused but there exists no species problem whereas in the model presented by Dino Keco et. al. (2012) most of operations performed has been moved from reduce phase to map phase. This change reduces amount of input output imprint because instead of HDFS all the processing data is kept in the memory, but species problem arises in this model as we have different population for each node.

In the third model an extension of Map Reduce is done with the addition of a second reduce phase and a special optimization on the merge phase is carried out for the second reduce operation. For handling heterogeneity and failures this extension of PGA on map reduce is beneficial.

The Hybrid approach proposed by Rohit Konkedar et. al. (2012) applies the local search algorithm for further optimizations and also generates initial population of very good quality. Large scale Vehicle Routing Problem with hundreds or thousands of customer nodes can be solved easily with very less time and full utilization of resource which is impossible in single machine.

The coarse grained model for the Automatic generation of Junit Test cases using PGA based Map Reduce proposed by Federica Sarro et. al. (2012) showed that using this technique on a small or less number of clusters the results obtained saved 50% of the using by using PGA and Map Reduce.
In the paper by Chao Jin et. al. (2008) on an Extension of Map Reduce for Parallelization of Genetic Algorithm it was observed that this extension benefits from the Map Reduce model on handling heterogeneity and failures.

### *5.2 Environment and Tools*

Table 1: Tools and Environments used in different papers for experimentation.

| | Name of the Author(s) | Year of publication | Processor | Operating System | No. of nodes Hadoop Cluster | RAM (in GB) |
|---|---|---|---|---|---|---|
| Scaling Genetic Algorithm using Map Reduce | A. Verma et. al. [11] | 2009 | Dual Intel Quad Core | Red Hat Enterprise | 52 | 16 |
| Parallelization of Genetic Algorithms using Hadoop Map/Reduce | Dino Keco et. al. [10] | 2012 | i7 Quad Core | Cent OS 5.6 | 10 | 4 |
| A parallel Genetic Algorithm Based on Hadoop Map Reduce for the Automatic Generation of Junit Test Suites | Federica Sarro et. al. [12] | 2012 | Intel Core i3 2100 | Windows 7 Home Premium SP1 64 bit | 3 | 4 |
| A Map Reduce based Hybrid Genetic Algorithm using Island Approach for solving Time Dependent Vehicle Routing Problem | Rohit Kondekar et. al. [14] | 2012 | Intel Core i3-2100 (Dual Core) | Ubuntu 11.04 | 7 | 3.1 |
| MRPGA : An Extension of Map Reduce for Parallelizing Genetic Algorithms | Chao Jin et. al. [13] | 2008 | Pentium 4 | Windows XP | 33 | 1 |

## 6. CONCLUSION

In this paper we reviewed some of the most recent representative publications on different models of parallel genetic algorithm using map reduce in the Hadoop platform for big data. This survey focused on the use of different models of parallel genetic algorithm for map reduce and the review started by giving an introduction on big data and the map reduce technology vastly used in big data and classified our work into three categories based on the three different parallel genetic algorithm models namely the global parallelization, coarse grained parallelization and the fine grained parallelization. Contributions by the associated authors in each of these categories using map reduce framework was discussed and the merits and demerits of each of the parallel model using map reduce was discussed and compared with each other. The use of different parallel models of genetic algorithm in

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

map reduce would help to analyze and execute the huge volume of data in big data efficiently in a parallel manner.

## REFERENCES

[1] Judith Hurwitz, Alan Nugent, Dr. Fern Halper, Marcia Kaufman, *Big Data for Dummies*, John Wiley & Sons, Inc., 2013, ch. 1.

[2] Md. Rezaul Karim, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, Ho-Jin Choi, "An efficient Distributed Programming Model for Mining Useful Patterns in Big Datasets", *IETE Technical Review*, vol. 30, pp. 53-63, 2013.

[3] Cairong Yan, Yongfeng Cairong, Guangwei XU, "Runtimes and Optimizations for Map Reduce", *IETE Technical Review*, vol. 30, pp. 506-515, 2013.

[4] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", *Communications of the ACM-50th anniversary issue: 1958-2008*, vol. 51, pp. 107-113, 2008.

[5] Xiao Yang, Jialing Sun, "An Analytical performance model of MapReduce*", IEEE Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 306-310, 2011.

[6] Frenzel,J.F, "Genetic Algorithms", *Potentials IEEE*, vol. 12, pp. 21-24, 2002.

[7] B.S.P. Mishra, S. Dehuri, "Parallel Computing Environments", *IETE Technical Review*, Vol. 28, Iss. 3, pp. 240-247, 2011.

[8] B.S.P. Mishra, S. Dehuri, R. Mall, A. Ghosh, "Parallel Single and Multiple Objectives Genetic Algorithms: A Survey", *International Journal of Applied Evolutionary Computation*, vol. 2, 2011.

[9] Erick Cantú-Paz, "A survey of Parallel Genetic Algorithms", *Calculateurs Parallelles, Reseaux et Systems Repartis 10(2)*, pp. 141 – 171, 1998.

[10] Dino Keco, Abdulhamit Subasi, "Parallelization of genetic algorithms using Hadoop Map/Reduce", *South East Europe Journal of Soft Computing*, 2012.

[11] Abhishek Verma, XavierLlor'a, David E. Goldberg, Roy H. Campbell, "Scaling Genetic Algorithms using MapReduce", *Ninth International Conference on Intelligent Systems Design and Applications*, *ISDA 09*, pp. 13-18, 2009.

[12] Linda Di Geronimo, Filomena Ferrucci, Alfonso Murolo, Federica Sarro, "A Parallel Genetic Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnit Test Suites", *2012 IEEE Fifth Conference on Software Testing, Verification and Validation*, pp. 785 – 793, 2012.

[13] Chao Jin, Vecchiola, C., Buyya, R., "MRPGA: An Extension of MapReduce for parallelizing Genetic Algorithms", *IEEE Fourth International Conference on eScience, 2008.eScience'08*, pp. 214-221, 2008.

[14] Rohit Kondekar, Akash Gupta, Gulshan Saluja, Richa Maru, Ankit Rokde, Parag Deshpande, "A MapReduce Based Hybrid Genetic Algorithm Using Island Approach for Solving Time Dependent Vehicle Routing Problem", *2012 International Conference on Computer and Information Science (ICCIS)*, vol. 1, pp. 263 – 269, 2012.