Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium, Volume 22, No. 1, ISSN 1726-9679
ISBN 978-3-901509-83-4, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2011
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2011

FRAMEWORK FOR DEVELOPMENT OF DISTRIBUTED EVOLUTIONARY ALGORITHMS BASED ON MAPREDUCE

PAVLECH, M[ichal]

Abstract: *The aim of this article is to introduce new framework for development of distributed evolutionary algorithms using MapReduce programming model. Its main goal is to provide researchers and developers with an easy solution for development of distributed optimization programs independent on evolutionary algorithms used*
Key words: *MapReduce, grid computing, distributed computing, hadoop, evolutionary algorithms*

1. INTRODUCTION

This paper is meant as an introduction to framework for simple development of evolutionary algorithms in distributed environment.

During development of software for evolution of artificial neural networks we have run into problems regarding computational cost of the software. Evolution of sufficient solution simply took too much time, even after its division into multiple threads on multi core and multi processor computers. In order to achieve acceptable run times, a decision was made to develop version of the software which could exploit distributed computing environment. Instead of a single purpose program a flexible framework was developed which allows adaptation of wide variety of evolutionary algorithms.

For the ease of development as well as previous experiences MapReduce programming model was chosen together with its implementation called Hadoop.

2. MAPREDUCE AND HADOOP

MapReduce is a programming model developed by Google (Ghemawat & Dean, 2004). Its original purpose was simplification of development of distributed applications for large scale data processing. The idea behind MapReduce is inspired by map and reduce functions in functional programming languages like LISP therefore developers require to define just two functions (or classes, depending on implementation), which serve as map and reduce part of resultant program.

For data storing and sharing, MapReduce is built with close relations to distributed file systems, optimized for large amounts of data and shared access. Important thing about data structure is that it needs to be in form of [key, value] pair.

In order to run MapReduce programs the framework servers have to be installed on a number of computers, which are usually of two types: distributed file system servers and actual computation nodes. One server is designated master node, which controls task assignments, monitors active nodes and cooperates all tasks and framework functions. MapReduce was developed with emphasis on its usage on commonly available hardware, where hardware and connection errors are likely to occur. To address these shortcomings MapReduce includes mechanisms which deal with failures. The simplest method is to check and restart – master periodically checks slave nodes, if a node does not respond it is added to a black list, so no new task are assigned to this node and failed task is reassigned to another node.

Several implementations of MapReduce exist, possibly the most popular and used open implementation is the Hadoop (Borthakur, 2007).

3. MAPREDUCE AND EVOLUTIONARY ALGORITHMS

There have been several attempts at using MapReduce model together with evolutionary algorithms (EA). EAs are incremental algorithms working in rounds, with output of each round is input for next round. In contrary, MapReduce is designed to run only once and produce final outputs immediately, this means that either MapReduce model of EAs have to be modified.

Probably the first reported usage of MapReduce for improving EAs was made in (Jin et al., 2007). They used custom MapReduce implementation, developed using .NET, called MRPGA, which included additional reduce phase to increase effectiveness of genetic algorithm (GA). Map function is used for fitness function evaluations, where each map function was called for one individual from population. 1st reduce phase chooses local optimum individuals and 2nd reduce selects global optimum individual which is emitted as final result. Mutation and crossover are implemented sequentially on master node. Above operations are repeatedly restarted with previous population as input until stopping criteria are met.

As a reply to previous work, Verma et al. presented another approach (Verma et al., 2009). Instead of developing their own MapReduce implementation they utilized Hadoop. They identified that Hadoop needs large amounts of time to initialize each map and reduce function, so in order to minimize this overhead, each map function processes more that just one individual. As was mentioned before, each map function gets data pairs of several individuals, where key represents genome and value represents fitness of individual, individuals are then evaluated and emitted from map functions. Each map function also selects and emits best individuals. Reduce functions randomly chose a number of individuals and perform crossover and emit new individuals as output. As in MRPGA, whole process is repeated until stopping criteria are met.

Another similar work which uses combination of GA and Hadoop is (Logofătu & Dumitrescu, 2011).

GA was not the only EA which was implemented via MapReduce model, other EAs include differential evolution (Zhou, 2010), which uses MapReduce for parallelization of fitness function evaluation, while other operations remain sequential (Tagawa & Ishimizu, 2010), which is implemented using threads instead of distributed architecture. And self-organizing migration algorithm (Pavlech, 2010), which uses MapReduce to parallelize mutation, crossover and evaluation in map functions without using reduce phase, but utilizes the distributed cache for distribution of “the leader” through the population. Particle swarm optimization (McNabb et al., 2007), this algorithm uses map function to update its position and velocity and reduce function updates global best values with information from other particles.

4. MRDEAF

All previous uses of MapReduce with EAs required a MapReduce process to be restarted once for each generation, in other words they used the global parallelization of EAs. Previous works indicated that framework overhead significantly decreased performance gain from adding new nodes (Verma et al., 2009; Pavlech, 2010). In order to decrease communication and initialization overhead, novel method for parallelization of EAs is proposed, called MRDEAF.

MRDEAF is an acronym for MapReduce Distributed Evolutionary Algorithms Framework and is developed on top of Hadoop.

MRDEAF uses more sophisticated idea than global parallelization, called coarse or fine grained parallel EAs (in some sources also island model). The whole population is divided into relatively isolated subpopulations, which evolve on their own, and uses migration operator which is used for moving individuals between subpopulations. Migrations are useful for increasing diversity of subpopulations, thus allowing them to explore wider search areas and increasing their chances of finding better solutions. Distinction between coarse and fine grained algorithms is in ratio of subpopulation count versus subpopulation size. Smaller count of bigger subpopulations (coarse grained) is best suitable for distributed architectures as it minimizes communication overheads, while vice versa solution is intended for massively parallel computers.

Island models introduce 5 additional parameters into EAs:

- Topology – determines to which subpopulations individuals migrate. High number of target islands may cause domination of one solution across population and premature convergence.
- Number of migrating individuals
- Migration rate – is migration occurs too often, algorithm is degraded to global parallelization.
- Method for selection of individuals for migration – if only best individuals migrate, they may draw population into local optimum and population will lose diversity.
- Method for replacement of individuals in target population

Detailed study of influence of topology as well as other parameters can be found in (Rucinski et al., 2010).

Main aim behind development of MRDEAF was to create framework, which would enable easy implementation of virtually any existing evolutionary algorithm into island models on distributed computing grid.

Another aim was to make framework as flexible as possible, user can define parameters of EAs for each subpopulation separately, it is possible to have completely different EA for each subpopulation, to change topology, migration methods etc. Creation of new algorithms, topologies, fitness functions, populations is done by simple subclassing of existing java classes.

Problem with using Hadoop is that for each map or reduce function call new java virtual machine (JVM) is started, this event considerably increases framework overhead. To minimize this overhead it was important to use as little map functions as possible. Each map function in MRDEAF is responsible for evolution of one subpopulation for a given number of generations. This way we are able to lower the number of JVM restarts. Implementation of migration between subpopulations relies on hdfs.

As was stated before, Hadoop requires its data to be in form of [key, value]. Value is a data structure which holds entire subpopulation (usually just a list of real valued arrays), while key is an integer number serving as a subpopulation label, migrating individuals are actually small subpopulations with migration bit enabled whose label determines the target population. Therefore defining new topology is fairly easy in principle, it involves creating function with two parameters: number of subpopulations and label of current subpopulation, and returning array of labels of target subpopulations. Using array enables migration into more than 1 subpopulation (for example hypercube topology).

5. CONCLUSION

MRDEAF undergone a series of tests which demonstrated its functionality. Test runs confirmed that for fitness functions with complexity high enough to benefit from distributed architecture and negate effects of long communication times.

Interesting feature was discovered during testing of the framework. By combining island architecture of EAs and flexibility of framework, it is possible to set different evolution parameters even different optimization algorithms to separate subpopulations. This opened opportunities for study of evolution dynamics based on characteristics and interactions of different algorithms used on different subpopulations.

Further research will be focused on this property of island models, as it is suspected that interactions between subpopulations and their alghorithms may be interpreted as a form of complex network and course of evolution better directed through detailed study of this network.

6. ACKNOWLEDGEMENT

This chapter was created as a part of internal grant agency project number IGA/40/FAI/11/D at Tomas Bata University in Zlin, Faculty of Applied Informatics and of the European Regional Development Fund under project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089

7. REFERENCES

Ghemawat, S., and Dean, J.: MapReduce: Simplified data processing on large clusters, *Proceedings of the 6th Symposium on Operating System Design and Implementation*, San Francisco, CA, USA, 2004

Jin, C., Vecchiola, C., and Buyya, R.: MRPGA: An extension of MapReduce for parallelizing genetic algorithms, *Fourth IEEE International Conference on eScience*, 214–221, 2008

Logofătu, D., and Dumitrescu, D.: Parallel evolutionary approach of compaction problem using mapreduce, *Parallel Problem Solving from Nature–PPSN XI*, Springer, 361–370, 2011

McNabb, A.W., Monson, C.K., and Seppi, K.D.: Parallel pso using mapreduce, *Evolutionary Computation*, 2007. CEC 2007. IEEE Congress on, 7–14, 2007

Pavlech, Michal: Distributed SOMA algorithm in MapReduce framework, *Proceedings of the International Masaryk conference*, 1380, 2010

Rucinski, M., Izzo, D., and Biscani, F.: On the impact of the migration topology on the Island Model, *Parallel Computing* 36(10-11), Elsevier, 555–571, 2010

Tagawa, K., and Ishimizu, T.: Concurrent Differential Evolution Based on MapReduce, *International Journal of computers*, 2010

Verma, A., Llora, X., Goldberg, D.E., and Campbell, R.H.: Scaling genetic algorithms using mapreduce, *Intelligent Systems Design and Applications*, 2009. ISDA'09. Ninth International Conference on, 13–18, 2009

Zhou, C.: Fast parallelization of differential evolution algorithm using MapReduce, *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 1113–1114, 2010