Parallel Genetic Algorithms on Cluster Architecture: A Case Study

Sisnett Hernandez, Ricardo.
sisnett@tesisinteractive.com

Abstract.

Genetic Algorithms have represented a powerful tool to solve optimization problems, mainly those with multivariate or complex objective functions. However due to the random nature of the algorithms, some search spaces appear too big to be covered, and the utility of GA´s seems diminished. Parallel Genetic Algorithms provide a way of increasing the coverage of search space with almost none drawbacks on performance or implementation difficulty. This paper presents the parallelization of a genetic algorithm to solve a multivariate combinatorial problem using message passing paradigm and a 16-node cluster. We explore two different ways of taking advantage of HPC: Increase in the search space and reduction of time using multiple processors. We also explore and explain some of the drawbacks of the cluster architecture on this applications and how to tackle them.

Keywords:
Genetic Algorithms, Evolutionary Computing, Genetic Programming, HPC

1. Introduction.

1.1 Genetic Algorithms

Genetic Algorithms (GA) are a search technique based on Darwin's 'survival of the fittest', where a group of solutions are ranked depending on their fitness, a numeric value which calculation is problem dependant created by John Holland. A population is created and evaluated then its individuals are reproduced via crossover and mutated to create new individuals to replace the least-fit individuals [1]. GA's have been proved useful in a variety of experiments, mainly used in multi-objective optimization where traditional (mathematic) techniques fail to find an answer either because they get stuck into a local minimum/maximum or because the amount of time needed to compute makes them useless.

1.2 Parallel Genetic Algorithms

Parallel Genetic Algorithms (PGA) are the distributed version of GA's. Using the apparent independence of diverse populations genetic algorithms can be parallelized almost in a trivial manner, and with very good results, this parallelization yields either an improvement in the time spent evaluating individuals or an increased coverage of the search space. There are two main versions of PGA's [2]:
Fine Grain Parallelism or Cellular Model: In this variant, each individual or a very small group of individuals is run in a different processing unit, and can only exchange genetic information (reproduce) with individuals in neighboring spaces.
Coarse Grain Parallelism or Island Model: This variant places more individuals per processing unit and exchanges information via a migration operator, this model appears more often when individual or generation evaluation can take a relatively long time.

1.3 Cluster Architecture

The concept of a cluster involves taking two or more computers and organizing them to work together to provide higher throughput, availability, reliability and scalability than can be obtained by using a single system [3].
Computers are connected using a high speed network interface (Gigabit Ethernet, Infiniband) and through a clustering software package they appear to the user as being just one entity with a lot of available resources.
For the experiments reported in this paper we used a 17 Node cluster, each node had 24 Gb Ram, 200 HDD @ 15000 RPM, Intel® Xenon® X7550 (8 Cores, 2Ghz), connected through a Gigabit Ethernet Switch. The clustering tool we used was

Rocks+® Clustering Software Package, and we ran on RedHat Enterprise Linux.

2. Implementation.

We implemented a traditional GA, using roulette for reproduction selection and 2 point cross for crossover; for parameters we used *mutation=0.15*, *reproduction=0.8* and *generation size=100* we ran for 100 generations, for all experiments the maximum possible fitness is 10,000.

For the parallel version, we chose coarse grain parallelism using each processor as an island and migrating after a fixed number of generations. For this we used the Message Passing Interface library (MPI) mainly because message passing paradigm models perfectly the idea of migration, adds very little overhead, is scalable and keeps blocking communication to a minimum. Migration occurred every 10 generations, and all populations migrated a random number of individuals in a random direction, this direction would be the same for this migration for all processors, i.e. all processors would migrate to the "EAST" between generation 10 and 11 and then all would migrate "NORTH" between generation 20 and 21.
At startup we would create a logical adjacency map based on the ids provided by MPI to each process. This grid-like map would then be turned into a torus to assure all processes have four neighboring processes with whom they could exchange individuals.

3. Experiments & Results.

For this paper we designed two main experiments that show the benefits of using cluster architecture with PGA's we denominated them *Time Attack* and *Search Space Increment* we present results and conclusions for each below.
3.1 Time Attack
For this experiment we increased the number of processors but kept the total amount of individuals, thus decreasing the population size as we added processors. This experiment showed a direct gain while adding processors, showing that the Amdalh's non-parallel section of GA's is considerably small when compared to the parallel section. Figure 1 shows the relation between linear speedup and our speed up.

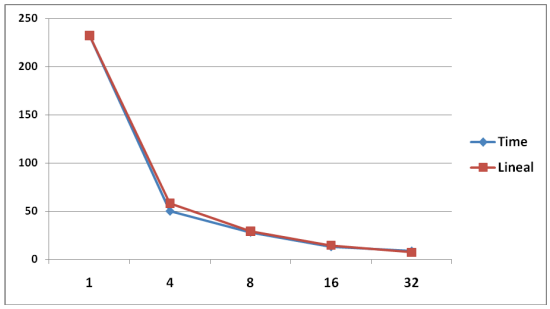

Figure1. Relation between time (in minutes) and the number of processes used to run the same population size.

On Figure 2 showing fitness vs number of nodes, we realize that the last case, for the one we used 32 nodes, we have a considerable decrease of fitness on the best individual, this problem arises due to the small size of population that led to a lack of genetic variability. However in the same chart we show improvements if we either increment the probability of mutation (Point A, mutation 35%) or if we increase the number of migrations during the experiment (Point B, 100% increase of migration).

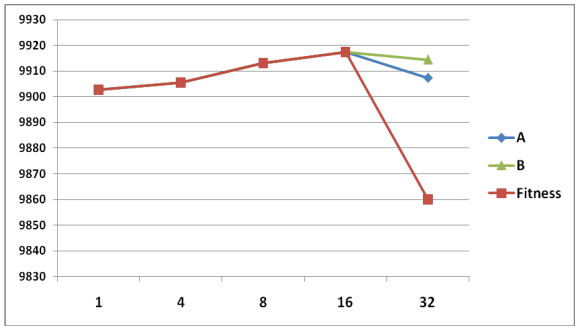

Figure 2. Relation between fitness and number of nodes used in the 'Time Attack' experiment, A represent the same experiment on 32 nodes but increased mutation and B with twice as much migration

3.2 Search Space Increment

For this experiment we increased the number of processors but kept the same amount of individuals for each one, experiment ran in the same time for all cases but overall fitness saw an increment while adding nodes. As shown on Figure 3 we can see that adding nodes increased the max fitness or the average fitness, which can be translated in a better understating of the search space.

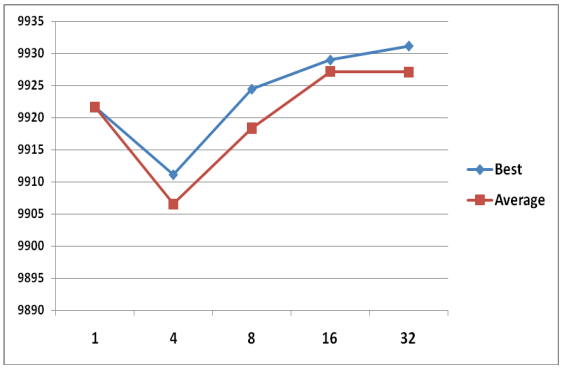

Figure 3. Relation between number of nodes and average and best fitness among nodes

3.2.1 An in-depth look into the 32 Node run.
For the biggest run of the search space coverage we present the final layout of the best individuals for each processor. In Figure 4 you can observe that populations with the best individuals tend to be together, while a line of under fitted populations can be seen (Sixth Row). This is one of the main issues of PGA's as good populations tend to be together or close leaving other populations under developed, this also translate as a loss in processing time since under-developed populations are less likely to provide a good answer but still are spending CPU and other resources.

Fitness Distribution

9919.34	9919.27	9919.27	9919.34
9923.07	9923.07	9919.27	9919.27
9923.1	9923.1	9918.73	9923.08
9923.11	9923.08	9919.01	9923.11
9923.08	9923.1	9918.67	9923.08
9917.18	9917.18	9917.18	9917.18
9919.48	9919.27	9919.27	9919.48
9919.48	9919.27	9919.27	9919.48

Figure 4. Distribution of fitness in the process geography. Processes on top row can migrate to the bottom row and vice-versa, this is also true for left-right most processes, generating a torus shape.

4. Issues & Future Work.

As mentioned before, the main problem with PGA's is the creation of clusters of underdeveloped populations, this ends up being ineffective as resources are spent in this populations which might not improve on the fitness of the most effective ones, this is especially true at the final stages of the GA.
Genetic Algorithms part from the premise that all genetic material is available at all times, and partitioning it yields different problems as we have seen. However, this is not necessarily true in the biological process in which GA's are supposed to be based; fittest individuals are not always present in every single population at the moment of mating. Our future work will center on solving this problem, either by current techniques or by proposing a new way.

Acknowledgements.

I would like to thank Intel® GDC for the processing time of its HPC cluster in which all this experiments were run. I would also like to thank PhD Marco Antonio de Luna from ITESM Campus Guadalajara for lending his code for MEWMA Optimization problem which we attacked with this PGA.

References.

[1] Holland J.H., *Adaptation in natural and artificial system,* Ann Arbor, The University of Michigan Press.

[2] Nowostawski, Mariusz. Poli, Riccardo. *Parallel Genetic Algorithm Taxonomy.*

[3] Guangzhon Sun. *A Framework for Parallel Genetic Algorithms on PC Cluster.* Proceedings of the 5th WSEAS Int. Conf. (pp274-278)

Cantu-Paz, Erick. *Topologies, Migration Rate and Multi-Population Parallel Genetic Algorithms.*

Guan, Yu. Xu, Baowen. Leung, Karl. *Parallel Genetic Algorithms with Schema Migration.*

Kazunori, Ishigame, Chakraborty, Hatsuo, and Makin. *Asynchronous Parallel Distributed Genetic Algorithm with Elite Migration.* International Journal of Information and Mathematical Sciences 4;2 2008.

Koza, Jhon. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.