Advances in Smart Systems Research : ISSN 2050-8662 : http://nimbusvault.net/publications/koala/assr/
Vol. 3. No. 3 : pp.49-54 : isrp13-005

Optimized Communications on Cloud Computer Processor by Using Parallel Genetic Algorithms and Parallel Computing

Nicolas Lassabe

ONERA - The French Aerospace Lab
F-31055, Toulouse, France
nicolas.lassabe@onera.fr

Abstract
Cloud computing and multi-core processors will change the way we program and use computer. This paper describes the benefits that cloud computing can obtain from optimized communications by parallel genetic algorithms. Chips with multi-core processors are very current and the performances of theirs applications depend a lot on their performances of communication between each process and applications. We show that how the performances could be easily multiplied by two with an optimization by parallel genetic algorithms. We applies this method for the Intel's single-chip cloud computer (SCC), a 48-core processor. This method could be applied to all cloud computers.

1. Introduction

The Intel's single-chip cloud computer (SCC) is a 48-core processor[1] [2]. The performances of the communications between the cores depends on theirs positions on the processor. For a same parallel application on the SCC, the distribution of the processes on the cores could impact the general performance[3]. The goal of this paper is to show the possibility of optimizing the communications by changing the positions of processes on the cores. For this, we use benchmarked applications by experts and meta-heuristic optimization methods like parallel genetic algorithms.

2. Background: The SCC's Communications

The cores on the SCC are distributed on a grid of 6x4 tiles (Fig 1). Each core is composed of two cores[1] [2]. The pipes of communication are also represented with a grid of 6x4 and each core is connected to one node of this grid. Each core can communicate directly with another core. During the communication between two cores, the message are send to the next tiles on the same row first, then finish on the tiles on the same column. If two cores are farer from each other on the grid, the communication will be slower. Also, the long communications have more probability that their pipes of communication are used by other cores at the same time[3].

Based on this expertise[3], we propose a fitness function f to evaluate the global performance of a parallelized application. For this, we measure the quantity of kB/s $q_{i,j}$ of data exchanged for each communication between two processes of this applications on different cores and we add the results to each nodes affected by the communications. We compare the quantity of data transited between each node $n_{i,j}$ of the grid (eq 1). The maximum of transited data by a node of the grid is the value of fitness (eq 2).

$$n_{i,j} = \sum_{c=0}^{nbcom} q_{i,j_c} \quad \text{(eq 1)} \qquad f = max\left(n_{i,j}\right) \quad \text{(eq 2)}$$

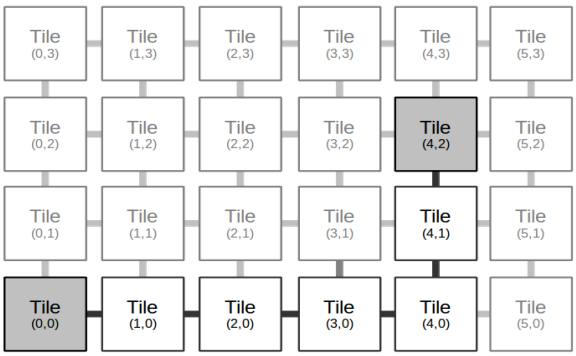


Fig 1: SCC's Architecture : a grid of 6x4 tiles composed of two cores . Example of communication from the title (0,0) to the tile (4,2). The cores (1,0),(2,0),(3,0), (4,0), (4,1) and (4,2) are affected by the communication. To communicate from the tile (4,2) to the tile (0,0), the cores (3,2),(2,2),(1,2),(0,2) and (0,1) are affected.

The challenge here is to have a better repartition of the processes in order to keep a good execution time and to minimize the fitness function (eq 1). To optimize the execution , we propose to have an uniform repartition of the processes on the grid. The communications between two processes on the same tile do not affect the performance of the communication. Moreover, we limit the number of processes by tiles to have a good execution and repartition of the processes on all the tiles.

3. The Parallel Genetic Algorithms

To optimize the results, we use parallel genetic algorithms[4] [5] [6] [7]. The Open MIP library[8] is used to exchange data among the processes. This library could use with all cloud computers and clusters.

The population of the genetic algorithms is composed of a list of positions of the processes on the cores of the SCC. Our genetic algorithm uses the model master-slave to parallelized the algorithm[4]. The master executes the main loop of genetic algorithm and sends the evaluation job of the fitness function to different slaves. Each slave receives a part of the population to evaluate and send back their results to the master.

The operators: A mutation function can change randomly the position of processes. A crossover function mixes two lists of applications with a random number cross-nodes.

To select the individuals of the population, we use tournament selection. Beside it, we use the fitness function previously describes to evaluate an individual (eq2).

4. Data and Experiments

For the experimentations, we use an application composed of 112 independent processes that communicate together. Each process can have a large quantity of data to exchange with different processes. This experimentation is from a real application used on the 48core SCC processors. Similar experimentations could be found in theses papers[3] [9]. The positions of the processes on the core change dramatically the performance (Fig. 5).

To optimize the evolution, we need to find the parameters that optimize the fitness function. The ideal percentage of selection population that gives good results is between 50% and 100%. For that of mutation should be between 1% and 100%. We run two hundred

experiments with 10 different values of selection and 20 different values of mutation distributed equitably. After these serial of experiments, we subtract the best fitness (the best fitness function was 3577, means 3577kB/s is maximal quantity of data exchange between two cores at the same moment) to the 200 fitness results (optimized values are between 3577 and 4822) (Fig. 2). We get results from 0 to 1245 after this operation of subtraction. These results shows the impacts of the parameters on the evolution. The size of population used was 48000 individuals.

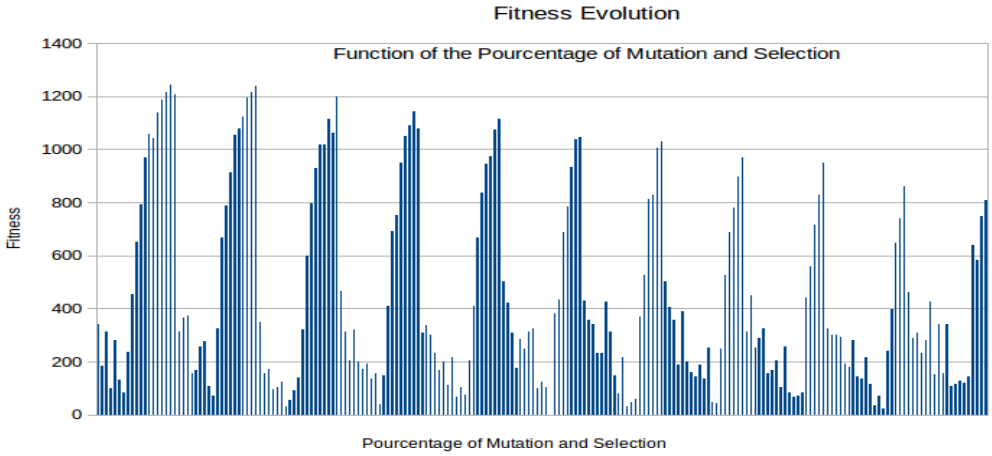


Fig 2: 200 experimentations to evaluate the influence of the mutation and selection parameters: the selection value starts from 50% to 100% by increment of 10. For each value of the selection value the mutation increments also of 5 from 1% to 100%. This show that the parameters are very important to have a good performances (diff 1245kB/s)

From previous experimentations, we obtain a couple of values for the selection and the mutation where the fitness function give good results (fitness under 3600, closed of 0 on the Fig 2). The figure 3 give the distribution of these couples of values.

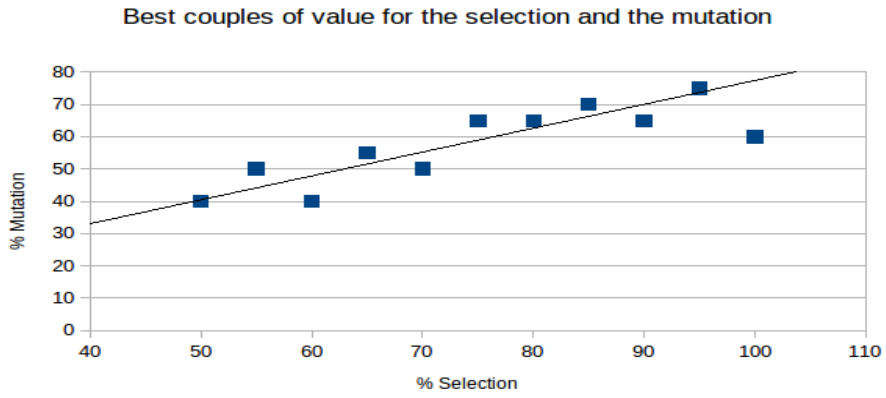


Fig 3: Couples of values for the selection and the mutation parameters where the fitness is under 3600.

5. Results

From the first results (Fig 2 and 3), we obtain that 75% of selection and 65% of mutation should converge for these parameters. With a size of population of 128000, we improve a little result of the fitness : 3573 (Fig 4) This experiment was done on a 48 core cluster.

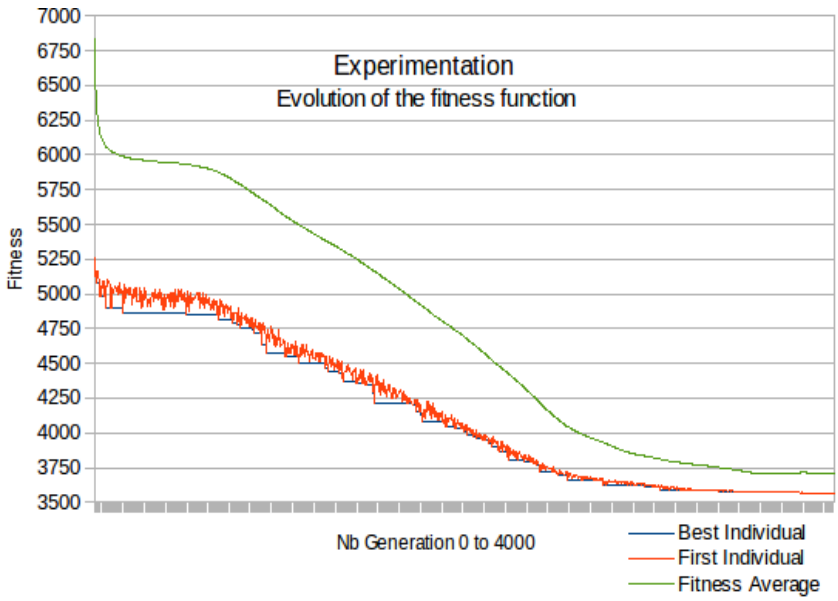


Fig 4: Evolution of the fitness function from generation 0 to 4000. The blue line is the evolution of fitness of the best individual. The red line is the evolution of the fitness of the best individual of the population (they are different from the blue line because we do not keep the best individual in the population). The green line is the evolution of the average fitness of the population.

To visualize the impact of the optimization of the position of the processes on a cloud computer like the SCC. We propose to display the quantity of data using the difference pipe of communication. On the figure 5.A, the positions of processes are random. We can see the pipes in black are larger. This means that a lot of data are transfered through these pipes and the communications are slower. On the figure 5.B, the same processes with optimized positions, the communication are more distributed and the communication is faster.

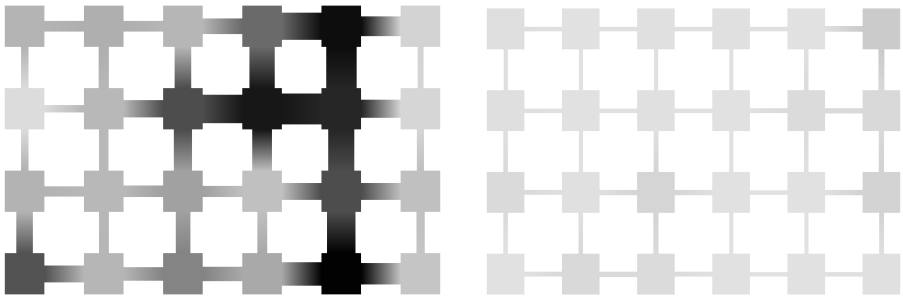
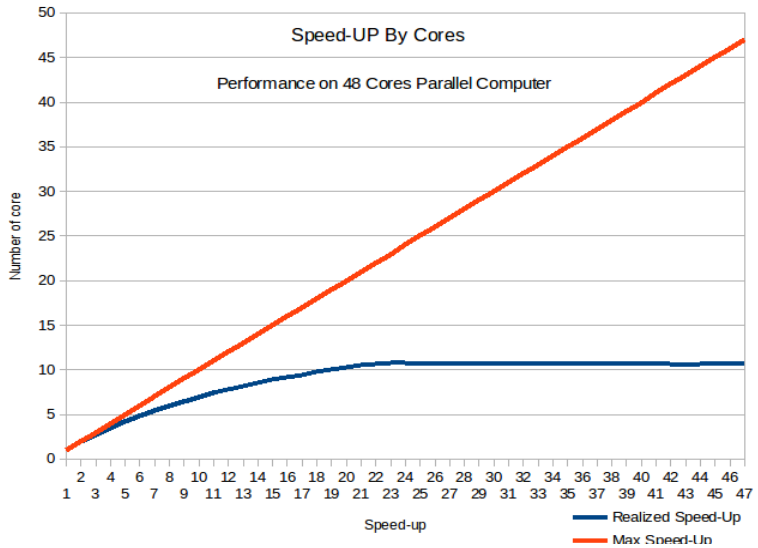


Fig 5A On the right, the communication of the SCC are not optimized. The exchanges of data are concentrated in the pipe in black (Fitness around 7000). Fig 5B On the left, the communication are optimized (Fitness 3573). The performance are two times better.

Notice the big difference of the size of pipes. This is because the pipes are much longer on the figure 5A. On the fig 5B the number of nodes that is affected by the communications are less. The communications are shorter and better optimized.



Our parallel genetic algorithm is executed on a 48 core cluster and the speed up is between 11 and 17. The process of optimization cloud be improved by compressing the size of data exchange between the master and the slaves.

Due to the big quantity of data, the speed-up we get on 48 cores cluster is between 11 and 17 (Fig. 3). To augment the performance of the parallel algorithms, we propose to compress the data sent from the master to the slaves. The data (the list position of processes) can be easily compressed.

6. Conclusions

The cloud computer will change the way we program[11]. This paper shows possibility to optimize the communication of the applications by using parallel computing and cloud computing by using parallel genetic algorithms. Also show how this method could be improved by selecting the good parameters.

The next step will be to improve the performances of the method by compressing the data exchange between the master and the slaves. It could be also interesting to see if it is possible to apply this method in real time. We may be able to identify the rules from our experiments to affect a better position directly in next step. We plan also to apply our method of optimization of the position of the processes on different cloud computer like the TilePro[11] to validate the concept.

Acknowledgements:
The author would like to thank Intel for granting an access to the SCC as part of the MARC program.

7. References

1. Intel Labs, “SCC external architecture specification (EAS),” Intel Corporation Tech. Rep., May 2010.
2. “The SCC programmer’s guide,” Intel Corporation, Tech. Rep., January 2012.
3. P. Gschwandtner, T. Fahringer, and R. Prodan, “Performance analysis and benchmarking of the intel SCC,” in Proceedings of the 2011 IEEE International Conference on Cluster Computing, ser. Cluster ’11. Washington, DC, USA: IEEE Computer Society, 2011
4. E. Cantu-Paz and D. E. Goldberg, “Efficient parallel genetic algorithms: Theory and practice,” in Computer Methods in Applied Mechanics and Engineering. press, 2000
5. E. Cant-Paz, “A survey of parallel genetic algorithms,” Calculateurs paralleles, vol. 10, 1998
6. M. Nowostawski and R. Poli, “Parallel genetic algorithm taxonomy,” in Proceedings of the Third International. IEEE, 1999
7. H. Mhlenbein, “Evolution in time and space - the parallel genetic algorithm,” in Foundation of genetic algorithms Morgan Kaufmann, 1991
8. The librairie Open MPI. Available: http://www.open-mpi.org/
9. W. Puffitsch, E. Noulard, and C. Pagetti, “Mapping a multi-rate synchronous language to a many-core processor,” in 19th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS’13), Philadelphia, Pennsylvania, USA, 2013.
10. Tile processor architecture overview for the Tilepro series, Tilera Corporation, 2013
11. Shekhar Borkar, Andrew A. Chien The Future of Microprocessors, Communications of the ACM, Vol. 54 No. 5, Pages 67-77 2011