The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

Asynchronous migration for parallel genetic programming on computer cluster with multi-core processers

Shingo KUROSE[1] Kunihito YAMAMORI[2] Masaru AIKAWA[3] and Ikuo YOSHIHARA[2]

[1]*Graduate School of Eng., Univ. of Miyazaki*
[2]*Faculty of Eng., Univ. of Miyazaki*
[3]*Technical Center, Faculty of Eng., Univ. of Miyazaki*
(Tel : 81 985 58 7589; Fax : 81 985 58 7589)
(kurose@taurus.cs.miyazaki-u.ac.jp,{yamamori,yoshiha,aikawa@cs.miyazaki-u.ac.jp)

Abstract: Island model is a typical implementation model of parallel distributed genetic algorithms, and it is also used in parallel genetic programming. Island model has migration process that exchanges individuals between sub-populations to leave local optimum. However island model requires synchronous process to exchange individuals at the same generation, and synchronous process increases computation time.

This paper proposes a new parallel genetic programming model based on the island model with asynchronous migration. We implement island model using Massage Passing Interface (MPI). Fitness calculation which requires the longest computation time is processed in parallel by multi-threading. In addition, proposed method employs a communication thread for migration between computation nodes, and communication thread communicates with another communication thread to exchange individuals at appropriate intervals. The communication and genetic operations can be independently processed on each core. Experimental results show that proposed method with five computation nodes and forty threads can reduce computation time about 17% of serial GP.

Keywords: Genetic programming, MPI, Multi-threading, Island model, Asynchronous migration

I. INTRODUCTION

Genetic Programming (GP) [1,2,3] is one of the evolutionary algorithms for optimization inspired from biological evolution. GP expresses a candidate of solution as a structural individual like a tree. Each individual is evolved by genetic operations such as crossover and mutation, and then only the individuals that have superior fitness remain for next generation. Through these evolutionary processes, GP can make a model automatically. Computation time of GP becomes longer as increasing of the number of individuals and the generations to obtain more accurate solution. So GP is usually implemented in computer cluster. Parallel GP implementation can be classified into two models; master-slave model and island model [2].

This paper proposes a new parallel genetic programming model based on the island model with asynchronous migration. We implement island model using Massage Passing Interface (MPI) [3]. Fitness calculation which requires the longest computation time is processed in parallel by multi-threading [4]. In addition, proposed method employs a communication thread for migration between computation nodes to reduce synchronous overhead. We evaluated proposed method on computer cluster.

II. PARALLEL GENETIC PROGRAMMING

1. Island model

Island model [2] is one of the implementation models for parallel distributed genetic algorithms and it is also used for parallel genetic programming. Island model divides a population consisting of individuals into sub-populations, and it assigns a sub-population to a computation node. Individuals in each sub-population

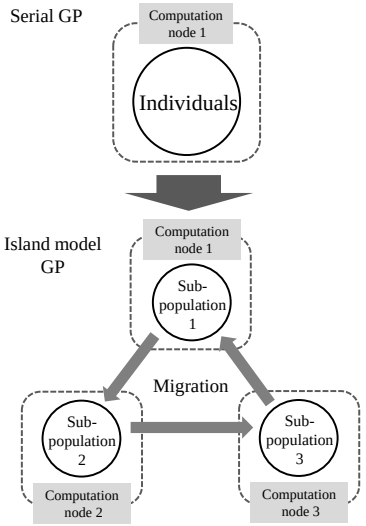

Fig. 1 An example of island model by three computation nodes.

are independently evolved in parallel. The number of individuals in sub-population is reduced, and genetic operations for individuals in each sub-population are processed in parallel, island model can reduce computation time. In usual, island model has migration process, that immigrates individuals from sub-population to another sub-population, to leave local optimum. Figure 1 shows an example of island model by three computation nodes.

2. Synchronous migration

Migration is a process to immigrate individuals from sub-population to another sub-population at every some generations. So the migration on computer cluster requires network communication to exchange individuals. Migrated individuals are chosen by various ways. For example, migrated individual is an elite that has the highest fitness in sub-population. In other way, some individuals are chosen at random. Before migration, computation nodes synchronize their generation because independent evolution leads the difference of computation time for each sub-population. So some sub-populations have to wait their migration until the slowest sub-population has finished their operations.

III. ASYNCHRONOUS MIGRATION FOR PARALLEL GENETIC PROGRAMMING

1. Inter-nodes parallelization and intra-nodes parallelization

Proposed method implements island model using Massage Passing Interface (MPI) [3] for inter-nodes parallelization. MPI is a standard library for parallel programming.

Recent processers equip some processing cores on a die, and they share a main memory. This architecture is suitable for multi-threading [4]. Our method creates threads for fitness calculation, and these threads work in parallel. Shared memory is used to exchange individuals between main thread and fitness calculation threads. Proposed method expects high speed processing by using both inter-nodes parallelization by MPI and intra-nodes parallelization by multi-threading. Figure 2 shows intra-nodes parallelization using fitness calculation threads.

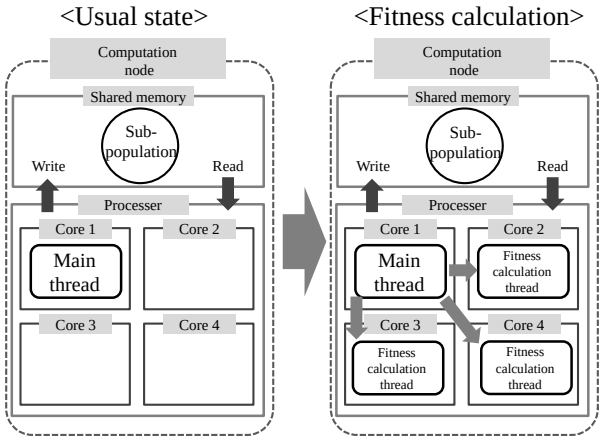

Fig. 2. Intra-nodes parallelization using fitness calculation threads.

2. Asynchronous migration

Proposed method creates a communication thread to reduce waiting time by synchronization. At first main thread creates individuals at random, and calculates their fitness. Then the main thread of each computation node selects an elite individual and stores it into the transmission buffer. Here, synchronization is executed just for once for reliable migration. Next, the main thread creates a communication thread and it transfer the individual stored in the transmission buffer to the communication thread in the other computation node at appropriate interval. The elite individual is exchanged through the shared memory between the main thread and the communication thread. The communication thread also receives the transferred individual from the other communication thread and stores it into the receiving buffer. The main thread takes the migrated individual from the other sub-population into their own sub-population at appropriate interval when the genetic operations are finished. Figure 3 shows an example of individual exchanging between the main thread and the communication thread. Figure 4 illustrates an example

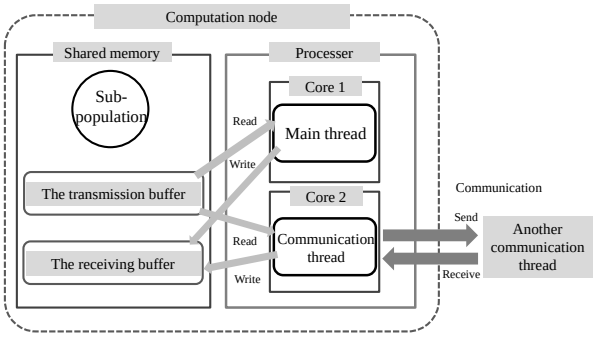

Fig. 3. An example of individuals exchanging between the main thread and the communication thread.

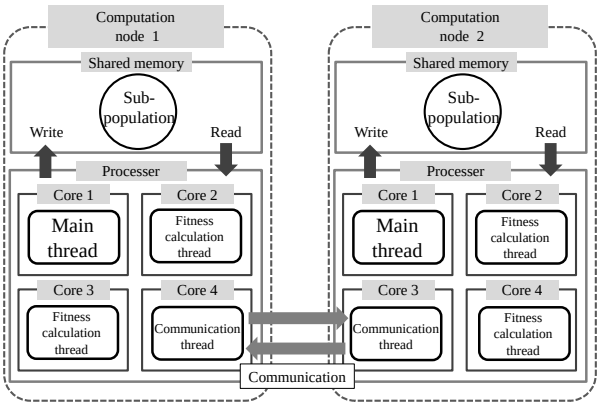

Fig. 4. An example of task allocation for two computation nodes with four processing cores in a processor.

of task allocation for two computation nodes with four processing cores in a processor.

IV. EXPERIMENTAL RESULTS

We evaluate processing time and accuracy of obtained solution on computer cluster. Table 1 shows experimental environments. Computer cluster equips distributed memory, and a multi-core processer has four processing cores, and it equips hyper threading technologies®, so eight threads can work in parallel. We compare with three implementation models of parallel GP. Table 2 shows detail of each model. The iGP is parallel GP using basic island model. The tfGP is the modified iGP using fitness calculation threads. In the tfGP, eight fitness calculation threads are created. The amGP is also the modified iGP using fitness calculation thread and communication thread for asynchronous migration. In the amGP, seven fitness calculation threads and one communication thread are created. We can evaluate the effect of intra-nodes parallelization by

Table. 1. Experimental environments.

CPU	Intel® Xeon E5530 2.40GHz ×2
Core/Thread	4 cores / 8 threads
Memory	8 GB
The number of computation node	12 nodes

Table. 2. Details of each model.

	iGP	tfGP	amGP
Island model	○	○	○
Threaded fitness calculation	×	○	○
Asynchronous migration	×	×	○

Table. 3. Parameters of GP.

Parameters	
The number of generation	1,000
The number of Individual	300
Max depth for Individual	8
The number of Training samples	600
Operator	+, -, *, /, sin, cos, tan, log
Selection	Elitist schemes Roulette selection
Mutation rate	0.1
Migrating individual	Elite
Migration interval	Every 50 generation

comparing the iGP with the tfGP. In addition, we can also evaluate the effect of asynchronous migration by comparing tfGP with amGP. Table 3 shows parameter of GP. Individuals are divided into sub-populations equally. In experimentation, training samples are taken from [5].

Figure 5 shows the computation time with respect to the number of computation nodes. Serial GP takes about 1,000 sec. for 1,000 generations. The iGP with five computation nodes can reduce computation time about 46% of serial GP. However, computation time is increased when more than six computation nodes are used. Figure 6 shows the average number of nodes in an individual for each number of computation nodes. In the iGP, the average number of nodes in an individual increases when the number of computation node is increased. Computation time of fitness calculation is increased by increasing of the number of nodes in an individual. As a result, the computation time for fitness calculation is different among sub-population, and it leads large overhead for synchronous migration. This is the reason of increasing of computation time when more than six computation nodes are used on the iGP and tfGP.

In Figure 5, the tfGP with five computation nodes and forty threads can reduce computation time about 21% of serial GP. Computation time is also increased when more than six processors are used as well as the iGP. It is because that synchronous migration takes large overhead as well as the iGP.

In Figure 5, the amGP with five computation nodes and forty threads can reduce computation time about 17% of serial GP.

Figure 7 shows the average fitness for each number of computation nodes. Fitness of individual s is

calculated by Equation (1).

$$Fitness = \frac{1}{F}$$
$$F = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \quad (1)$$

In Equation (1), n is the number of training samples, $\hat{y}_i$ and y_i are the desired value and the output of the model for the i-th training sample, respectively. In Figure 7, the average fitness of amGP increases as well as the iGP and the tfGP when the number of computation node is increased. In Figure 5, computation time of amGP doesn't increase even if more than six processors are used. So, proposed method is effective method when many computation nodes are used to obtain more accurate solution.

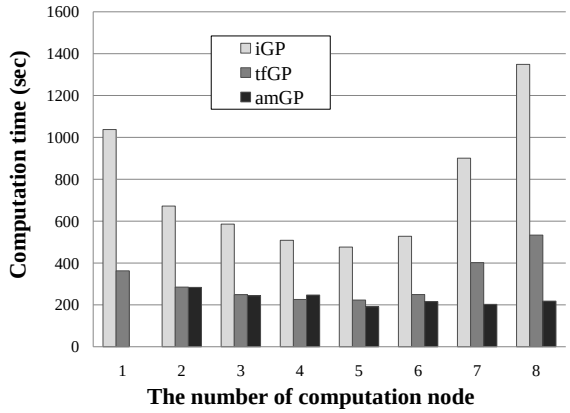

Fig. 5. Computation time with respect to the number of computation nodes.

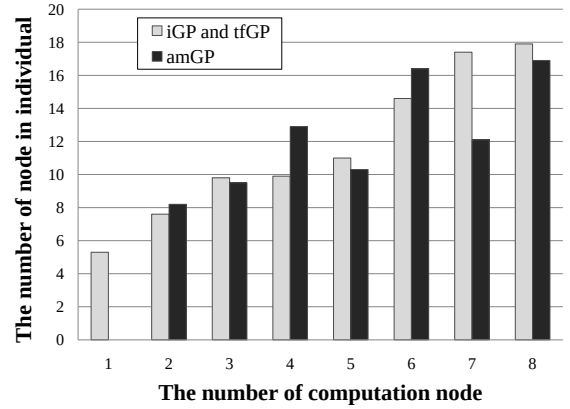

Fig. 6. Average number of nodes in individuals for each number of computation nodes.

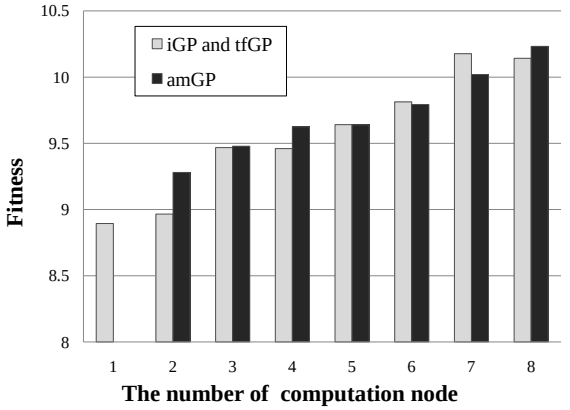

Fig. 7. Average fitness for each number of computation nodes.

V. CONCLUSION

This paper proposed a new parallel genetic programming model based on the island model using asynchronous communication between computation nodes. We implement island model using Massage Passing Interface (MPI). Our method creates threads for fitness calculation, and these threads work in parallel. In addition, proposed method employs a communication thread for asynchronous migration between computation nodes. Experimental results showed that fundamental parallel island model with five processors can reduce computation time about 46% of serial GP. Furthermore, our proposed method with five computation nodes and forty threads can reduce computation time about 17% of serial GP. In future, we will improve efficiency of parallelization by investigating the load of each computation node.

REFERENCES

[1] Numata N, Sugawara K, Yamada S et al. (1999), Time Series Prediction Modeling by Genetic Programming without Inheritance of Model Parameters. Proc. Fourth Int. Sym. Artificial Life and Robotics:500-503

[2] Eklund SE (2003), Time Series Forecasting using Massively Parallel Genetic Programming. Parallel and Distributed Processing Symposium: 10.1109/IPDS.2003.1213272

[3] Messom CH and Walker MG (2002), Evolving Cooperative Robotic Behaviour using Distributed Genetic Programming. Control, Automation, Robotics and Vision: 215-219

[4] Nichols B, Buttlar D, Farrell JP (1996), Pthreads Programming (A Nutshell handbook). Oreilly & Associates Inc (California).

[5] Kamiguchi M, Yamamori K, Yoshihara I et al. (2009), An Automatic Model Building for Screening Functional Foods with GP. ICROS-SICE International joint Conference 2009:3679-3684