ISSN 1870-4069

Use of Parallel Patterns of Communication between Processes for search of Sequences DNA and RNAi Strings

Mario Rossainz-López[1], Manuel Capel-Tuñón[2], Ivo Pineda-Torres[1], Ivan Olmos-Pineda[1], Arturo Olvera-López[1]

[1]Benemérita Universidad Autónoma de Puebla, Faculty of Computer Science, Puebla, Mexico

[2]University of Granada, Software Engineering Department, College of Informatics and Telecommunications, Granada, Spain

{rossainz,ipineda,iolmos, aolvera}@cs.buap.mx, manuelcapel@ugr.es

Abstract. Within an environment of Parallel Objects, an approach of Structured Parallel Programming and the paradigm of the Orientation to Objects show a programming method based on High Level Parallel Compositions or HLPC to solve two problems of combinatorial optimization: grouping fragments of DNA sequences and the parallel exhaustive search (PES) of RNA strings that help the sequence and assembly of DNAs in the construction of gnomes. The Pipeline and Farm models of parallel patterns of communication between processes are shown as HLPC under the Object Orientation paradigm to solve the cited problems. Each HLPC contains a set of predefined synchronization constraints between processes, as well as the use of synchronous, asynchronous and asynchronous future modes of communication. We show the algorithms that solves the problems, their design and implementation as HLPC and the performance metrics in their parallel execution using multicores and video accelerator card.

Keywords. HLPC, parallel structured programming, parallel objects, pipeline, farm, DNA, RNAi.

1 Introduction

The parallel processing is an alternative to the sequential processing when the limit of performance of a system is reached. In the sequential computation a processor only carries out at the same time an operation, on the contrary of what happens in the calculation parallel, where several processors they can cooperate to solve a given problem, which reduces the time of calculation since several operations can be carried out simultaneously. The present research uses structured parallel programming through a POSIX thread library as a methodological programming proposal based on the pattern of the High-Level Parallel Compositions HLPC [1], [2], the which it is based on the paradigm of Orientation to Objects to solve problems parallelizable using parallel objects.

In this work supply a library of classes that provides the programmer the communication/interaction patterns more commonly used in the parallel programming: the patterns pipeline and farm. With them, problems like the assembly of DNA strings and Parallel Exhaustive Search (PES) of RNAi strings proposed in this paper can be solved. In the case of the first problem mentioned, it is a problem of combinatorial optimization in which diverse heuristics and met heuristics have been proposed to assemble sequences of DNA strings and to provide essential information to understand the species and their mechanisms of life including the human species.

This work shows the implementation of a grouping algorithm that evaluates a set of DNA sequence fragments as a HLPC. The HLPC represents a Farm where worker processes are themselves Pipeline HLPCs. The algorithm determines subgroups of fragments by DNA sequences matching found, which have a high probability of being aligned in an assembly task. Each worker process of HLPC Farm works in parallel with the other worker processes that are generated with a group of fragments of DNA sequences that are internally constructed as graphs represented through the HLPC Pipeline and through an in-depth search the new groups of DNA sequences are generated, which must be processed by some assembly technique to form the contigs of a genome that has been sequenced covering most of its structure but missing a fragment to be completed.

Finally, the design of an experiment is shown through the use of the new HLPC generated called HLPC GraphADN, with genomes of viruses and bacteria available on the web. The pseudo random synthetic readings created to form contigs are shown and the execution performance of this proposal is obtained for eight genomes with an Intel Core i8 processor, a video accelerator card with 1664 CUDA cores and a clock frequency of 1178 MHz. In the case of the second problem a Parallel Exhaustive Search (PES) of RNAi strings is implemented using the inter-process communication pattern called FARM through an HLPC. The authors propose a new HLPC model called HLPC ARNi that is based on the HLPC Farm, which is part of a parallel object library whose details can be consulted in [3], [4].

The HLPC RNAi performs a pre-processing with the input data through its controller object of the Farm, which consists in join in a single string, all the input RNAi strings including the string that contains the characteristics of the organism that is analyzed. The controller then distributes to the farm worker objects within the HLPC RNAi the string constructed so that they, in parallel, perform the Parallel Exhaustive Search, [5], [6], find the matches based on a pre-established substring length and return the search results.

An experiment was designed using the HLPC ARNi with the RNAi strings database located in the Pombase site, referring to the yeast species known as S. pombe., with different lengths of substrings for the search of RNAi strings of the species of yeast mentioned and results of execution times and performance analysis were obtained using 3 to 8 cores of a dual-core Intel server machine.

2 Definition of High Level Parallel Compositions (HLPC)

Using an OO-programming environment, the basic idea is implementing any type of parallel communication patterns between the processes of an application or distributed/parallel algorithm. A HLPC comes from the composition of a set three object types: an object manager (Fig. 1) that represents the HLPC itself and makes an encapsulated abstraction out of it that hides the internal structure [7].

The object manager controls a set of objects references, which address the object collector and several stage objects and represent the HLPC components whose parallel execution is coordinated by the object manager. The objects stage are objects of a specific purpose, in charge of encapsulating a client-server type interface that settles down between the manager and the slave-objects. And a collector object, we can see an object in charge of storing the results received from the stage objects to which is connected, in parallel with other objects of HLPC composition. During a service request the control flow within the stages of a HLPC depends on the implemented communication pattern.

Manager, collector and stages are included in the definition of a PO [8]. POs are active objects, which have intrinsic execution capability. Applications that deploy the PO pattern can exploit the inter-object parallelism as much as the intra-object parallelism. A PO-instance object has a similar structure to that of an object in C++, and additionally defines a scheduling policy that specifies the way in which one or more operations carried out by the instance synchronize. The communication modes used are: The synchronous communication, the asynchronous communication and the asynchronous future. The Synchronization policies are expressed in terms of restrictions; for instance, mutual exclusion in reader/writer processes or the maximum parallelism allowed for writer processes.

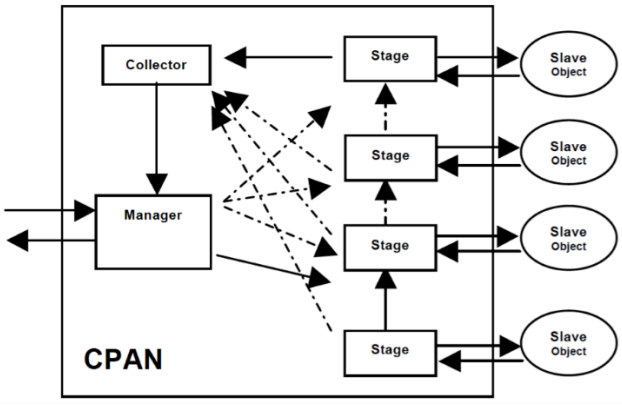


Fig. 1. Internal structure of HLPC.

3 The HLPC Pipeline

We can use this parallel processing technique to solve a problem by splitting it into a series of successive tasks so that data flow in the direction given by the process interconnection structure. Each task can therefore be completed sequentially [9]. In a pipeline, each task is executed by a processor or process as shown in Fig. 2. Using the technique of the pipeline, the idea is to divide the problem in a series of tasks that must be completed, one after another, see Fig. 2. In a pipeline each task can be executed by a process, thread or processor for separate, [9].

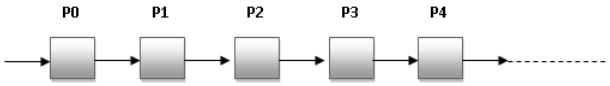


Fig. 2. Internal structure of HLPC.

The parallel pipeline processing technique is presented as a high-level parallel composition for solving a wide variety of problems from several different areas that can already be solved by partially sequential algorithms. In this way, the HLPC pipeline guarantees code parallelization of the resulting algorithm while taking advantage of existing sequential algorithms by using the HLPC processing pattern. The Fig. 3 represents the parallel pattern of communication Pipeline as a HLPC. The details of the implementation can be consulted in [10].

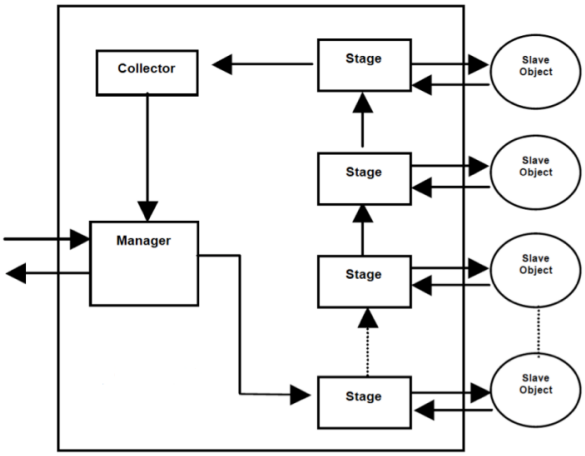


Fig. 3. The HLPC of a Pipeline.

4 The HLPC FARM

The technique of the parallel processing of the farm as a HLPC is shown here. The so named farm parallel pattern of interaction is made up of a set of independent processes,

called worker processes, and a process that controls them, called the process controller [11]. The worker processes are executed in parallel until all of them reaches a common objective. The process controller distributes the work and controls the progress of the farm until the solution of the problem is found [12]. Fig. 4 shows the pattern of the farm.

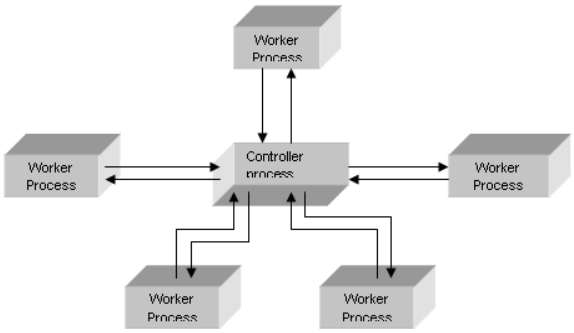


Fig. 4. Farm with a controller and five workers.

The representation of parallel pattern farm as a HLPC is shown in Fig. 5. The details of the implementation can be consulted in [13].

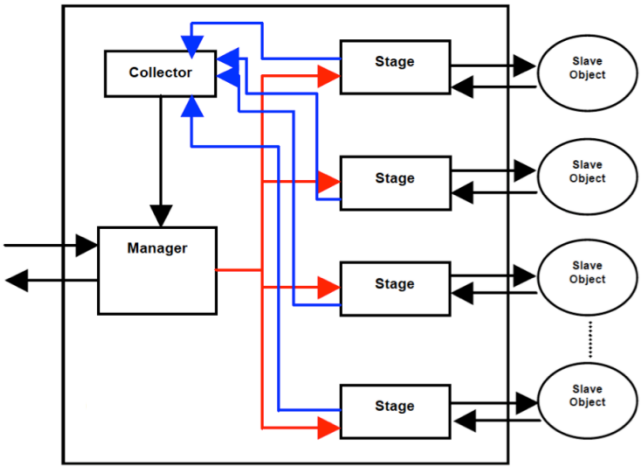


Fig. 5. The HLPC of a farm.

5 Methodology

1. The slave objects, which are executed by the stages and represent instances of the functionality required from the HLPC are created.

2. The sequential algorithm that solves the initial problem is implemented and its instances become methods associated to the slave objects.
3. A list of associations (e.g. slave object, associated method) are created.
4. An instance of the Manager is created to represent the HLPC-manager being constructed. It is then initialized with the list of associations from the previous step. At this moment, the necessary internal stages of the HLPC-manager are automatically created as instances of the Stage and each obtains an association.
5. The initial data to be processed are specified by creating data types and user-defined data as objects, whereas the input data set represents the problem to solve.
6. Application execution is requested to find a solution to the problem.
7. Parallel execution is achieved by transparently using synchronous, asynchronous and future asynchronous communication for the user's main program. The HLPC then creates its Collector object, which is passed as a reference to every stage connected with the Collector object, together with a copy of the data set to be processed. The initial stage works with these data and they are processed by the slave object associated to each stage. The result is then passed on to the next stage which is then created and initialized by the previous stage. In the same way, the second stage processes the data received when its slave object method is executed, and the new result is sent to the next stage, and so on. Finally, the obtained results are received by the Collector object which then compiles, processes and sends them to the Manager object which, in turn, sends the solution outside the HLPC.
8. Finally, the results are shown to the user.

6 Case Study 1: Parallel Exhaustive Search of RNA Strings using HLPC

The search for DNA strings for the construction of genomes of an organism is included within the genomics area and the representation of their information within bioinformatics is done digitally through formats used for this purpose, which are in general flat text formats, relatively simple and easy to analyze (parsing) associated with a search algorithm. The best known and which is used in the present work is the gene sequence representation format called FASTA, [14]. Today there are different approaches for this type of searches, but the most common is to use parallel computing technologies due to the large number of data that are generated and that must be processed in order to achieve an assembly of acceptable DNA sequences. In this work, it is proposed by using the HLPC model to carry out a Parallel Exhaustive Search (PES) of RNA or DNA strings using the communication pattern called FARM. The PES was carried out in plain text files containing a representation of RNA strings of an organism with its respective name. In the HLPC Farm used, the process controller or manager performs the pre-processing of the file extracting the strings written in the FASTA format to create the dictionary formed of the characteristics of the strings and the strings themselves, which is sent to each worker process or stage to perform the exhaustive search using the associated algorithms in the manager object and in the slave objects of the HLPC. This search is carried out in parallel by all the farm worker processes. The HLPC Farm

always guarantees a workload balance of these procsesses thanks to the synchronization restriction of the maximum parallelism that its components have guaranteeing the reduction in the execution times of each worker process, but also of the HLPC Farm itself. Then a new model of HLPC called HLPC ARNi is created, which is shown in Fig. 6. The pre-processing of the data is done through a text file as input to the HLPC ARNi. This file contains the name of a text string, as well as its characteristics, which is sent to the Manager of the HLPC or Farm Controller Process. The Manager has the Pre-Processing through a Slave Object, which consists of joining in a single string, all the ARNi strings that are in the TXT input file including the line text that contains the characteristics of the organism in question, then the Manager distributes to each Stage process the corresponding workload defining the limits start and end of the PES for each Stage.

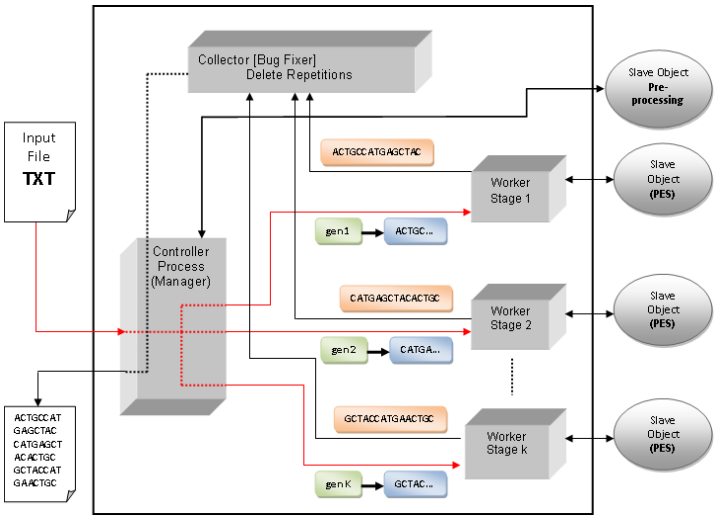


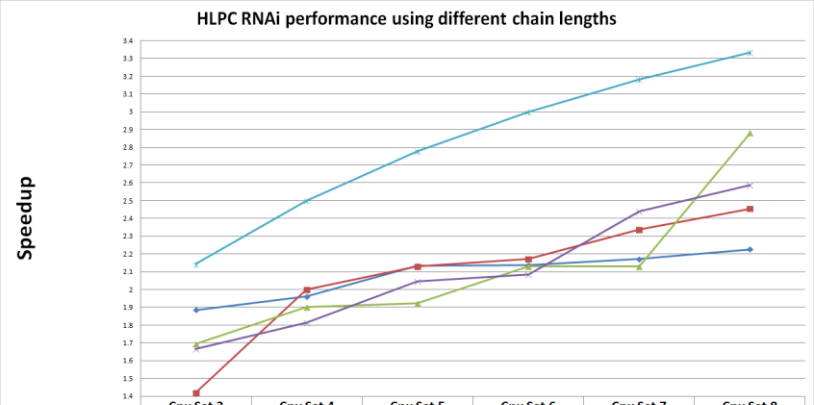
Fig. 6. Model of HLPC ARNi.

The load balance is made using the maximum parallelism in each Stage process (worker) of the Farm which is based on the identifier number of each worker process, [15], [16]. Subsequently the PES is performed in each stage of the HLPC ARNi executing the associated algorithm through the corresponding Slave Object and if a predefined substring is found within the ARNi string it is sent to the Collector object, which receives them in parallel from all the stage processes (workers) connected to it. The Collector bugs fixer eliminates repetitions of strings and the result is sent to the Manager who in turn sends it to an output file to the user.

6.1 Efficiency of HLPC ARNi

An experiment was designed using the HLPC ARNi with the RNAi string database located at the Pombase site (ftp://ftp.ebi.ac.uk/pub/databases/pombase/pombe/Chro-

mosome_Dumps/fasta/). Easts are the agents of fermentation and are found on the surface of plants [17]. The experiment was carried out on a server machine with Intel Xeon processor 2630 2.40 GHz and 8 cores. The operating system installed is Linux Ubuntu 64 bits, with a RAM of 125 Gbytes and a clock frequency of 1200 MHz. Experiments were performed with different number of nodes to determine if the workload was performed correctly. To determine if the length of the strings to be searched has a direct impact on the execution time, experiments with different string lengths were performed for their search, where the execution times increase according to the length of the search string grows, but on the other hand the execution times decrease as the number of nodes (nuclei) that are used in the execution of the search increases. In Fig. 7 shows the scalability of the Speedup found in the HLPC RNAi for different string lengths using 3 to 8 nodes in its execution, showing generally a good acceleration as the number of nodes increases.



	Cpu Set 3	Cpu Set 4	Cpu Set 5	Cpu Set 6	Cpu Set 7	Cpu Set 8
Chain Length = 10	1.885336273	1.961009174	2.132169576	2.1375	2.172808132	2.2265625
Chain Length = 20	1.418485237	2	2.129094412	2.173058014	2.338624339	2.455555556
Chain Length = 30	1.69481654	1.901960784	1.92333113	2.128749086	2.128749086	2.881188119
Chain Length = 40	1.665427509	1.813765182	2.0456621	2.08372093	2.440087146	2.586605081
Amdahl	2.142857143	2.5	2.777777778	3	3.181818182	3.333333333

Fig. 7. Scalability of the magnitude of the Speedup found for the HLPC RNAi in exclusive nodes of 2, 3, 4, 5, 6, 7 and 8 cores.

7 Case Study 2: Assembly of DNA Sequences using HLPC

DNA sequencing is the process of determining the precise order of nucleotides within a DNA molecule [18]. Knowledge of DNA sequences has become essential to carry out basic biological research. In this regard use of HLPCs for grouping DNA sequence fragments from the parallelization of a clustering algorithm to evaluate a set of fragments are made, which have a high probability of being aligned in an assembly task [19]. The algorithm finds the splices between the fragments using the Myers algorithm and links them in a graph. Then an in-depth search is done in the graph to form the groups and send them as a result. The assembly of DNA strings is proposed as a com-

binatorial optimization problem and is classified as NP-hard and is based on the paradigm divide-and-conquer using a structure type farm, so that the computational cost of finding the sequence alignments and its splice is substantially reduced with respect to its sequential version. The number of processes required to process the fragments of DNA sequences of a specific genome such as that of a virus or bacteria is determined by the splice of the strings found by the sequential solution algorithm, which looks in parallel for overlaps in the remaining fragments. Two sub-strings of each fragment are taken for comparison with other fragments; and thus, splices are located and associated with the processes (a process for each splice sequence of DNA strings). A splice graph is then generated that shows the relationship between pairs of nodes (processes), as well as the lack of communication among others. The set of nodes (processes) of the graph that are inter-related are grouped together within a worker process pattern farm. Each set of related nodes in the graph are independent and represent the grouping of fragments found. In Fig. 8 is shown the representation of HLPC for grouping DNA sequence fragments.

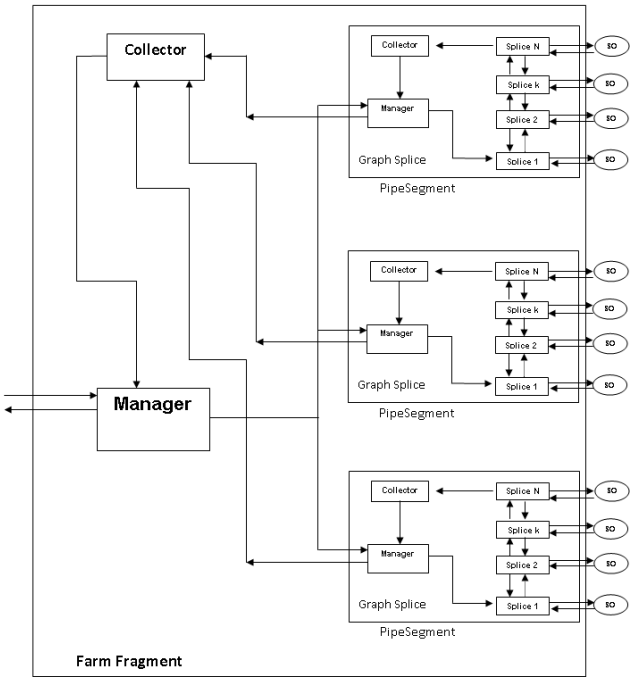


Fig. 8. The HLPC GraphADN.

The new HLPC named HLPC GraphADN is structured as a FARM of n-worker processes, i.e., n-fragments of DNA sequences and each worker process is itself a two directions-communication pipeline HLPC formed by m-stages where each stage of HLPC Pipe represents a splice sequence of DNA strings connected with both, the previous stage as the next stage. The collector object receives the number of formed groups and the elements that belong to each of the formed groups. With the latter information

collected, an in-depth search is performed to locate these items and obtain the sequence groups formed by the sequential algorithm assigned to each of the HLPC's slave objects with this result, the user can use an assembly of DNA sequences to try to complete a particular genome or to finish an incomplete sequence of DNA strings of some animal or plant type species.

7.1 Efficiency of HLPC GraphADN

An experiment was designed by using the HLPC GraphADN with genomes of viruses and bacteria available on the web whose data were obtained from European Nucleotide Archive, http://www.ebi.ac.uk. The Scalability of the speedup found in HLPC GraphADN is shown in Fig. 9 to 12. The plot shows the number of processes deployed for the calculation of eight genomes in an experiment conducted on a computer Intel Core i8 processor and using a video accelerator card with 1,664 CUDA cores.

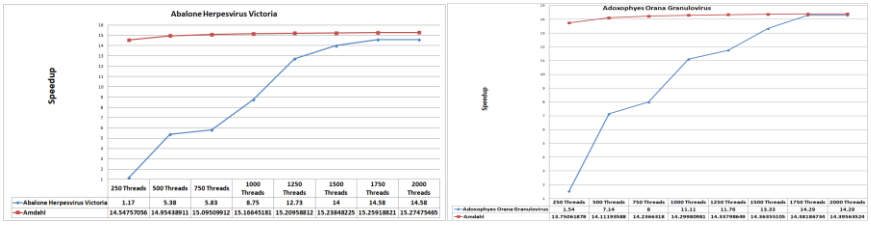


Fig. 9. Speedup found for the HLPC GraphADN for the genome Abalone herpesvirus Victoria and Adoxophyes orana granulovirus.

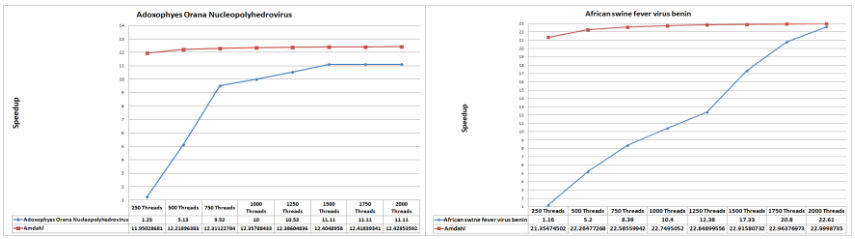


Fig. 10. Speedup found for the HLPC GraphADN for the genome Adoxophyes orana nucleopolyhedrovirus and African swine fever virus benin.

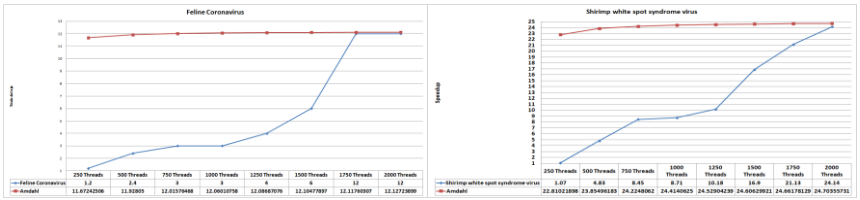


Fig. 11. Speedup found for the HLPC GraphADN for the genome Feline coronavirus and Shirimp white spot syndrome virus.

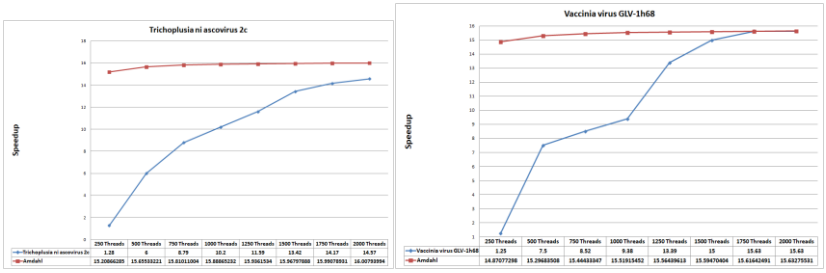


Fig. 12. Speedup found for the HLPC GraphADN for the genome Trichoplusia ni ascovirus 2c and Vaccinia virus GLV-1h68.

8 Comparative Analysis with Other Parallel Approaches

In the literature there are parallel approaches to obtaining DNA sequences that run on parallel machines or computers with multiple cores and that show the obtained accelerations as results. Such are the cases of references [20], [21] and [22]. In [20] the parallelization of two bioinformatic applications used in the analysis of long DNA sequences is shown. Its parallel proposal shows good acceleration results in computers with several processors but not in multicore computers. In [21] the same authors of the reference [20] show the accelerations found in the parallel analysis of DNA sequences using GPU and CUDA. Their results for different sizes of DNA sequences show problems to obtain good acceleration performances and propose to develop new techniques in data processing. Finally, in [22] describe parallel algorithms with different complexities that exhaustively determine all words of size k, k being arbitrarily large, in a source DNA sequence. The results shown that our algorithms achieve a high degree of scalability, allowing the detection of DNA words of 64 nucleotides in only 800 seconds.

Comparing the performance and acceleration of our proposal with those cited, it is concluded that the use of the HLPC in the two case studies presented shows better results than those published in [20] and [21] but not with the results of [22] that are better.

9 Conclusions

The authors have presented a method of design concurrent applications based on the HLPC construction that has been shown susceptible to be used in different platforms. The authors discuss the implementation of HLPCs pipeline and farm as generic and reusable patterns of communication/interaction between processes, which can even be used by inexperienced parallel application programmers to obtain efficient code by only programming the sequential parts of their applications. The HLPC pipeline and HLPC farm have been extracted from a larger library of parallel objects is made up of six sets of classes that constitute the implementation of the parallel patterns farm, pipe and others like treeDV as HLPCs that are part of what will be the library of High Level Parallel Compositions. The authors have presented two case studies: the parallel exhaustive

search of RNAi strings through the new HLPC RNAi constructed; referring to the yeast species known as S. pombe, and parallel calculation of the DNA sequences for 8 genomes. In both cases of study, the efficiency of the HLPCs in the solution of the problems has been shown. It has been presented the speedup and low execution times w.r.t. best sequential version of the algorithms that solve these problems. The authors have also obtained good performance in their executions and speedup scalability (compared to Amdahl's law) on the number of processors used to obtain the solution of complex problems.

References

1. Corradi A., Leonardi L.: PO Constraints as tools to synchronize active objects. Journal Object Oriented Programming 10, pp. 42-53 (1991)
2. Danelutto, M.; Orlando, S; et al.: Parallel Programming Models Based on Restricted Computation Structure Approach. Technical Report-Dpt. Informatica, Universitá de Pisa (1995)
3. Rossainz, M., Capel M.: A Parallel Programming Methodology using Communication Patterns named HLPCS or Composition of Parallel Object. In Proceedings 20TH European Modeling & Simulation Symposium.Campora S., Giovanni, Italy (2008)
4. Rossainz, M., Capel M.: Compositions of Parallel Objects to Implement Communication Patterns. In Proceedings XXIII Jornadas de Paralelismo, SARTECO 2012, Elche, España (2012)
5. Blelloch, Guy E.: Programming Parallel Algorithms. Comunications of the ACM, Volume 39, Number 3 (1996)
6. Kumar, Vipin, et al: Introduction to parallel computing. Design and Analysis of Algorithms., USA, Benjamin-Cummings (1994)
7. Rossainz, M.: Una Metodología de Programación Basada en Composiciones Paralelas de Alto Nivel (HLPCs). Universidad de Granada, PhD dissertation (2005)
8. Corradi A, Leonardo L, Zambonelli F.: Experiences toward an Object-Oriented Approach to Structured Parallel Programming. DEIS technical report no. DEIS-LIA-95-007 (1995)
9. De Simone, et al.: Designs Patterns for Parallel Programming. PDPTA International Conference (1997)
10. Rossainz M, Capel M., Domínguez P.: Pipeline as high level parallel composition for the implementation of a sorting algorithm. In Proceedings 27TH European Modeling & Simulation Symposium.Campora Bergeggi, Italy (2015)
11. Roosta, S.: Parallel Processing and Parallel Algorithms. Theory and Computation, Springer (1999)
12. Barry W., Allen M.: Parallel Programming. Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall, ISBN 0-13-671710-1 (1999)
13. Rossainz M., Capel M.: Approach class library of high level parallel compositions to implements communication patterns using structured parallel programming. In Proceedings 26TH European Modeling & Simulation Symposium.Campora Bordeaux, France (2014)
14. Pearson W.R., Lipman D.J.: Improved tools for biological sequence comparison. In Proceedings of the National Academy of Sciences of the United States of America 85, http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml (1988)
15. Lee R.C.T., Tseng S.S., Chang R.C., Tsai Y.T.: Introducción al diseño y análisis de algoritmos, un enfoque estratégico. Mc Graw Hill (2007)
16. Levitin A.: The Design of Analysis of Algorithms. Wesley (2003)

17. Wood V., Gwilliam R., Rajandream M.A., et al.: The genome sequence of Schizosaccharomyces pombe. Nature (2003)
18. Masoudi-Nejad, A., Narimani, Z. and Hosseinkhan, N.: Next Generation Sequencing and Sequence Assembly. SpringerBriefs in Systems Biology (2013)
19. DanishAli, S. and Farooqui, Z.: Approximate Multiple Pattern String Matching using Bit Parallelism: A Review. International Journal of Computer Applications, Volume 74, No.19, 47–51 (2013)
20. Sanjuan A., Arnau V., Claver J.M.: Análisis Paralelo de Secuencias AND sobre computadores con multiples cores. Actas de las XIX Jornadas de Paralelismo, Castellón, España (2008)
21. Peña A.J., Claver J.M., Sanjuan A., Arnau V.: Análisis Paralelo de Secuencias AND mediante el uso de GPU y CUDA, ResearchGate. https://www.researchgate.net/publication/228857228 (2014)
22. Yang X.Y., Ripoll A., Marin I., Luque E.: Genomic-scale analysis of DNA Words of Arbitrary Length by Parallel Computation. Parallel Computing: Current & Future Issues of High-End Computing, NIC Series, Vol. 33, 623–630 (2008)