Proceedings of the World Congress on Engineering 2010 Vol I
WCE 2010, June 30 - July 2, 2010, London, U.K.

Distributed Memory Implementation of the False Nearest Neighbors Method: Kd-tree approach versus Box-assisted approach

Julio J. Águila[1], Ismael Marín[2],Enrique Arias[1],
María del Mar Artigao[2], Juan J. Miralles[2] *

Abstract—

In different fields of science and engineering (medicine, economics, oceanography, biological systems, etc) the False Nearest Neighbors (FNN) method has a special relevance. In some of these applications, it is important to provide the results in a reasonable time scale, thus the execution time of the FNN method has to be reduced. To achieve this goal, a multidisciplinary group formed by computer scientists and physicists are collaborative working on developing High Performance Computing implementations of one of the most popular algorithms that implement the FNN method: based on Kd-tree data structure and based on Box-assisted algorithm. In this paper, a comparative study of the distributed memory architecture implementations carried out in the framework of this collaboration, is presented. As a result, two parallel implementations for Box-assisted algorithm and one implementation for the Kd-tree structure are compared in terms of execution time, speed-up and efficiency. In terms of execution time, the approaches presented here are from 2 to 19 times faster than the sequential implementation, and the kd-tree approach is from 2 to 6 times faster than the box-assisted approaches.

Keywords: ***Nonlinear Time Series Analysis, False Nearest Neighbors method, Kd-tree data structure, Box-assisted method, message passing interface***

1 Introduction

In nonlinear time series analysis the false nearest neighbors (FNN) method is crucial to the success of the subsequent analysis. Many fields of science and engineering use the results obtained with this method. But the complexity and size of the time series increases day to day and it is important to provide the results in a reasonable time. For example, in the case of electrocardiogram study (ECG), this method have to achieve real-time performance in order to take aome prevension actions. With the development of the parallel computing, large amounts of processing power and memory capacity are available to solve the gap between size and time.

The FNN method was introduced by Kennel et al. [1]. Let $X = \{x(i) : 0 \leq i < n\}$ a time series. We can construct points (delay vectors) according to Eq.(1):

$$y(i) = [x(i), x(i+\tau), \ldots, x(i+(d-1)\tau)] \qquad (1)$$

where τ is the embedding delay and d is the embedding dimension (Fraser and Swinney [2]). The Takens embedding theorem [3] states that for a large enough embedding dimension $d \geq m_0$, the delay vectors yield a phrase space that has exactly the same properties as the one formed by the original variables of the system. The FNN method is a tool for determining the minimal embedding dimension d. Working in any dimension larger than the minimum leads to excessive computation when investigating any subsequent question (Lyapunov exponents, prediction, etc.).

The method identifies the nearest neighbor $y(j)$ for each point $y(i)$. According to Eq. (2), if the normalized distance is larger than a given threshold R_{tr}, then the point $y(i)$ is marked as having a false nearest neighbor.

$$\frac{|x(i+d\tau) - x(j+d\tau)|}{||y(i) - y(j)||} > R_{tr} \qquad (2)$$

Eq. (2) has to be calculated for the whole time series and for several dimensions $d = 1, 2, \ldots, m$ until the fraction of points, which must be lower than R_{tr}, is zero, or at least sufficiently small (in practice, lower than 1%).

While greater is the value of n (length of the time series), the task to find the nearest neighbor for each point is more computationally expensive. A review of methods to find nearest neighbors, which are particularly useful

*This work has been supported by National Projects CGL2007-66440-C04-03 and CGL2008-05688-C02-01/CLI. Albacete Research Institute of Informatics[1] and Applied Physcis Dept.[2], University of Castilla-La Mancha, Avda. España s/n,02071-Albacete, Spain, email for correspondence:{Enrique.Arias}@uclm.es, Telephone number: +34-967-599200 Ext: {2497}, Fax number: +34-967-599224.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

for the study of time series data, can be found in [4]. We focused in two approaches: based on the box-assisted algorithm, developed in the context of time series analysis by Grassberger [5]; and the based in a kd-tree data structure [6, 7] developed in the context of computational geometry.

According to Schreiber [4], for time series that have a low dimension of embedding (e.g. up to the 10's), the box-assisted algorithm is particularly efficient. This algorithm can offer a lower complexity of $O(n)$ under certain conditions. By the other hand, accordingly with the literature if the dimension of embedding is moderate an effective method for nearest neighbors searching consists in using a kd-tree data structure [6, 7]. From the computational theory point of view, the kd-tree-based algorithm has the advantage of providing an asymptotic number of operations proportional to $O(n \log n)$ for a set of n points, which is the best possible performance for arbitrary distribution of elements.

We have applied the paradigm of parallel computing to implement three approaches directed towards distributed memory architectures, in order to make a comparative study between the method based on the box-assisted algorithm and the method based on the kd-tree data structure. The results are presented in terms of performance metrics for parallel systems, that is, execution time, speed-up and efficiency. Two case studies have been considered to carried out this comparative study. A theoretical case study which consists on a Lorenz model, and a real case study which consists on a time series belonging to electrocardiography.

The paper is organized as follows. After this introduction, a description of the considered approaches is introduced on Section 2. On section 3, the experimental results are presented. Finally, on Section 4 some conclusions and future work are outlined.

2 Method

We selected two programs to start this work: the `false_nearest` program based on the box-assisted algorithm [8, 9]; and the `fnn` program based on a kd-tree data structure [10].

We employ the paradigm *Single-Program, Multiple Data* (SPMD) [11] to design the three parallel approaches. A coarse-grained decomposition [12] has been considered: we have a small number of tasks in parallel with a large amount of computations. The approaches are directed towards distributed memory architectures using the Message Passing Interface (MPI [13]) standard for communication purpose. Two approaches are based on the box-assisted algorithm and the another approach is based on the kd-tree data structure.

2.1 Approaches Based on Box-assisted Algorithm

The box-assisted algorithm [5] considers a set of n points $y(i)$ in k dimensions. The idea of the method is as follows. Divide the phase space into a grid of boxes of side length ϵ. Each point $y(i)$ lies into one of these boxes. The nearest neighbors there are located in the same box or in one of the adjacent boxes. The `false_nearest` program is a sequential implementation of the FNN method based in this algorithm.

By profiling the `false_nearest` program in order to carry out the parallel approaches, four tasks were identified. Let X a time series, Y a set of points constructed according to Eq. (1), `BOX` an array that implements the grid of boxes (or mesh), and p the number of processes. Two parallel implementations were formed based on these four tasks:

Domain Decomposition Time series X is distributed to the processes. Two ways of distribution have been developed: Time Series (TS) and Mesh (M). In a TS data distribution the time series is splitted into p uniform parts, of length $\frac{n}{p}$, being n the length of the time series. In a M data distribution, each process computes the points that lie in its range of rows. The range of the mesh rows is assigned by $\frac{s}{p}$, where s is the size of the BOX.

Grid Construction The `BOX` array is filled. Two ways of grid construction have been developed: S (Sequential) and P (Parallel). In a S construction each process fill the BOX sequentially, thus each one has a copy. In a P construction each process fills a part of the the group of boxes located over a set of assigned mesh rows.

Nearest Neighbors Search Each process solves their subproblems given the domain decomposition way. In a TS data distribution each process use the same group of points Y. In a M data distribution each process can use different groups of points.

Communication of Results Processes use MPI to synchronize the grid construction and to communicate the partial results at the end of each dimension.

The approaches were called follow the next nomenclature: `DM-P-M` meaning a **D**istributed **M**emory implementation considering that the grid construction is in **P**arallel and the time series is distributed according to the **M**esh; `DM-S-TS` meaning a **D**istributed **M**emory implementation considering that the grid construction is **S**equential and the **T**ime **S**eries is uniformily distributed to the processess.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

2.2 Approach Based in the kd-tree Data Structure

A kd-tree data structure [6, 7] considers a set of n points $y(i)$ in k dimensions. This tree is a k-dimensional binary search tree that represents a set of points in a k-dimensional space. The variant described in Friedman et al. [7] distinguish between two kinds of nodes: internal nodes partition the space by a cut plane defined by a value of the k dimensions (the one containing a maximum spread), and external nodes (or buckets) store the points in the resulting hyperrectangles of the partition. The root of the tree represent the entire k-dimensional space. The `fnn` program is a sequential implementation of the FNN method based in this structure.

`fnn` program has been also analyzed by means of a profile tool before making the parallel implementation, identifying five main tasks. Thus, let X a time series, n the length of the time series, Y a set of points constructed according to Eq. (1), `KDTREE` a data structure that implement the kd-tree, p the number of processes, and $q = \{1 \dots p\}$ a process identifier. For convenience we assume that p is a power of two. The parallel implementation called `KD-TREE-P` was formed based on these five tasks:

Global kd-tree building The first $\log p$ levels of `KDTREE` are built. All processors perform the same task, thus each one has a copy of the global tree. The restriction $n \geq p^2$ is impose to ensure that the first $\log p$ levels of the tree correspond to nonterminal nodes instead of buckets.

Local kd-tree building The local `KDTREE` is built. In the level $\log p$ of the global tree are p nonterminal nodes. Each processor q build a local kd-tree using the qth-node like root. The first $\log p$ levels are destroyed and `KDTREE` is pointed to local tree.

Domain Decomposition Time series X is distributed to the processess. The building strategy impose a distribution over the time series. Thus, the time series is split according to the kd-tree algorithm and the expected value of items contained in each local tree is approximately $\frac{n}{p}$.

Nearest Neighbors Search Each process solves their subproblems. Each process search the nearest neighbors for all points in Y that are in the local `KDTREE`.

Communication of Results Processes use MPI to communicate theirs partial results at the end of whole dimensions. The master process collects all partial partial results and reduces them.

3 Experimental results

In order to test the performance of the parallel implementations, we have considered two case studies: the Lorenz time series generated by the equations system described in 1963 by E.Lorenz [14]; the electrocardiogram (ECG) signal generated by a dynamical model introduced in 2003 by McSharry et al. [15]. The Lorenz system is a benchmark problem in nonlinear time series analysis and the ECG model is use for biomedical science and engineering [16].

The parallel implementations have been run in a supercomputer called GALGO, which belongs to the Albacete Research Institute of Informatics [17]. The parallel platform consists in a cluster of 64 machines. Each machine has two proccesors Intel Xeon E5450 3.0 GHz and 32 GB of RAM memory. Each proccesor has 4 cores with 6144 KB of cache memory. The machines are running RedHat Enterprise version 5 and using an Infiniband interconnection network. The cluster is presented as a unique resource which is accessed through a front-end node.

The results are presented in terms of performance metrics for parallel systems described in Grama et al. [12]: execution time T_p, speed-up S and efficiency E. These metrics are defined as follows:

- **Execution Time:** The serial runtime of a program is the time elapsed between the begining and the end of its execution on a sequential computer. The parallel runtime is the time that elapses from the moment that a parallel computation starts to the moment that the last processing element finishes its execution. We denote the serial runtime by T_s and the parallel runtime by T_p.
- **Speed-up** is a measure that captures the relative benefit of solving a problem in parallel. It is defined as the ratio of the time taken to solve a problem in a single processing to the time required to solve the same problem on a parallel computer with p identical processing elements. We denote speed-up by the symbol S.
- **Efficiency** is a measure of the fraction of time for which a processing element is usefully employed; it is defined as the ratio of speed-up to the number of processing elements. We denote efficiency by the symbol E. Mathematically, it is given by $E = \frac{S}{p}$

Let p the number of processors, the execution time of the approaches have been tested for $p = \{1, 2, 4, 8, 16, 32\}$, where $p = 1$ correspond to the sequential version of the approaches. We used one million records of the time series to calculate the ten first embedding dimensions, and obtaining that the optimal time delay for Lorenz time series is $\tau = 7$ and for ECG signal is $\tau = 5$.

In order to obtain the best runtime of the approaches based in a box-assisted algorithm we found the best size of `BOX` for each value of p (tables 1 and 2). The size of

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

`BOX` define the number of rows and columns for the grid of boxes. The values for $p = 1$ correspond to the sequential version of the program `false_nearest`.

p	DM-P-M	DM-S-TS
1	8192	8192
2	4096	4096
4	2048	4096
8	2048	4096
16	2048	2048
32	2048	2048

Table 1: Size of `BOX` for each value of p using a Lorenz time series.

p	DM-P-M	DM-S-TS
1	4096	4096
2	4096	4096
4	4096	4096
8	2048	4096
16	2048	2048
32	2048	2048

Table 2: Size of `BOX` for each value of p using a ECG time series.

We have run 10 tests to obtain the median value of the execution time T_p. In total 360 tests were performed. The performance metrics results are showing in Figs.[1-2].

Sequential kd-tree implementation shows a lower execution time than box-assisted approach, since the grid construction stage on box-assisted implementation in TISEAN is very expensive in terms of execution time.

The behaviour of the Lorenz case study and the ECG case study is quite similar. Notice that, according to Figures 1.b and 2.b, it is possible to appreciate a super-linear speed-up for kd-tree implementation when $p < 8$ and these performance decreases when $p > 8$. The super-linear speed-up is explained due to the fact that the cache memory is better exploited and that when the tree is splitted less searches have to be done at each subtree. With respect to the lost of performance, this situation is produced due to different causes. The first one is that, evidently, the overhead due to communications increases. Also, the most importat cause is that the sequential part of our implementation becomes every time more relevant with respect to the parallel one.

DM-S-TS is the box-assisted implementation that provides the best results for the Lorenz attractor and the ECG signal. The reason is the very best data distribution with regard to DM-P-M. However, the reconstruction of the mesh is not parallelized in DM-S-TS implementation. So, the sequential part makes the reduction of execution

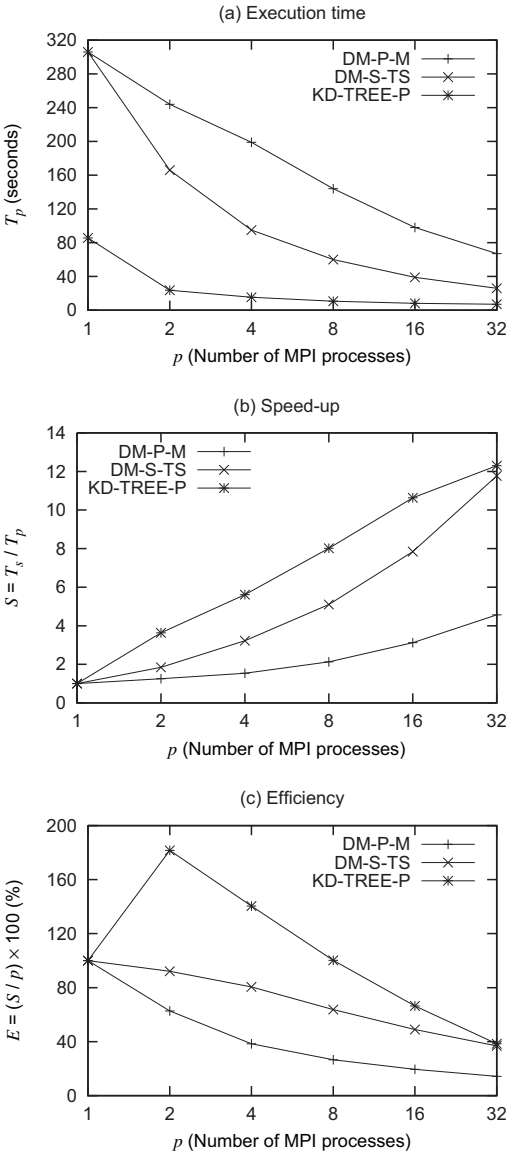

Figure 1: Performance metrics for the Lorenz case study: (a) Execution time, (b) Speed-up and (c) Efficiency

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

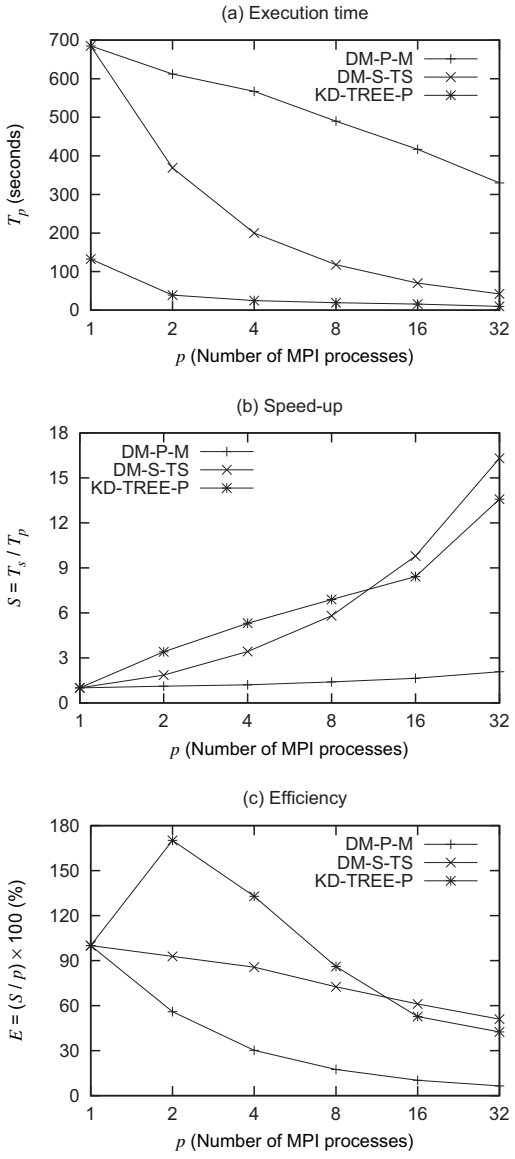

Figure 2: Performance metrics for the ECG case study: (a) Execution time, (b) Speed-up and (c) Efficiency

time less significant when more CPUs are used. However, as the execution time of find neighbors is increased (e.g. in larger times series data), this circumstance becomes very less important.

For Lorenz attractor, the DM-S-TS implementation is around 1.8 faster than the sequential program when it uses 2 CPUs, and around 12 when it uses 32 CPUs. This means that the efficiency for 2 CPUs is around 92% and decreases to 37% when using 32 CPUs. As Fig. Y shows, the best box-assisted parallel implementation achieves a speed-up of around 16 when it is run on 32 CPUs of GALGO. Moreover, the time saving is around 94% using 32 CPUs, and even using only 4 CPUs the time saved is around 71%. Unlike previous cases, the efficiency of best implementation decreases more slowly.

An optimization of TISEAN has been used. It allows the best mesh size to be tuned for each case. In case of use original TISEAN (fixed mesh size), the reduction of execution time would be more important.

According to the experimental results, kd-tree-based parallel implementation obtains the best performance than the box-assisted-based parallel implementation, almost in terms of execution time, for both cases studies. Due to the spectacular execution time reduction provided by the kd-tree-based parallel implementation, the performance in terms of speed-up and efficiency seems to be worst, with respect to the other approaches.

4 Conclusions

In this paper, a comparative study between the distributed memory implementations of two different ways to compute the False Nearest Neighbors method have been presented, that is, the based on the box-assisted algorihtm and the based on kd-tree data structure. To make this comparative study three different implementations have been developed: two implementations based on box-assited algorithm, and one implementation based on kd-tree data structure.

The most important metric to consider is how well the resulting implementations accelerate the compute of the minimal embedding dimension, which is the ultimate goal of the FNN method. In terms of the execution time, the parallel approaches are from 2 to 19 times faster than the sequential implementation, and the kd-tree approach is from 2 to 6 times faster than the box-assisted algorithm.

With respect to the experimental results, the kd-tree-based parallel implementation provides the best performance in terms of execution time, reducing dramatically the execution time. As a consequence, the speedup an efficiency are far from the ideal.

About related works, in the context of parallel implementations to compute FNN method, the work carried out

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

by the authors could be considered as the first one. The authors are working also on considering shared memory implementations (Pthreads [18, 19] or OpenMP [20, 21]) and hybrid (MPI+Pthreads or MPI+OpenMP) parallel implementations.

Also, as a future work, the author are considering to develope GPUs-based parallel implementation of the algorithms considered in this paper.

However, it is necessary to deal with more case studies of special interest for the authors: wind speed, ozone, air temperature, etc.

To sum-up, we hope that our program will be useful in applications of nonlinear techiques to analyze real time series as well as artificial time series. This work represents the first step of nonlinear time series analysis, that it is becomes meaningful when considering ulterior stages on the analysis as prediction, and when for some applications the time represents a crucial factor.

References

[1] M.B. Kennel, R. Brown, and H.D.I. Abarbanel. Determining Embedding Dimension for Phase Space Reconstruction Using the Method of False Nearest Neighbors. *Physics Review A*, 45(6):3403–3411, 1992.

[2] A.M. Fraser and H.L. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33(2):1134–1140, 1986.

[3] Takens, F. (1981). Detecting strange attractors in turbulence. *In Rand, D.A. and Young, L.-S. (eds.) Dynamical Systems and Turbulence, Warwick 1980, Springer: New York, pp. 366-381.*

[4] T. Schreiber. Efficient neighbor searching in nonlinear time series analysis. *Int. J. Bifurcation and Chaos*, 5:349, 1995.

[5] P. Grassberger. An optimized box-assisted algorithm for fractal dimensions. *Physics Letters A*, 148(1-2):63–68, 1990.

[6] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[7] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[8] R. Hegger, H. Kantz, and T. Schreiber. Practical implementation of nonlinear time series methods: The TISEAN package. *Chaos*, 9(2):413–435, 1999.

[9] R. Hegger, H. Kantz, and T. Schreiber. Tisean: Nonlinear time series analysis. *http://www.mpipks-dresden.mpg.de/tisean*,[27 Aug 2009], 2007.

[10] M.B. Kennel. Download page of `fnn` program. *ftp://lyapunov.ucsd.edu/pub/nonlinear/fns.tgz*,[24 May 2007], 1993.

[11] F. Darema. The spmd model: Past, present and future. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 1–1, 2001.

[12] A. Grama, Gupta A., Karypis G., and Kumar V. *Introduction to Parallel Computing*. Addison-Wesley New York, 2003.

[13] Message Passing Interface. *http://www.mcs.anl.gov/research/projects/mpi/*,[27 Aug 2009].

[14] E.N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[15] P.E. McSharry, G.D. Clifford, L. Tarassenko, and L.A. Smith. A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomedical Engineering*, 50(3):289–294, 2003.

[16] Ecgsyn: A realistic ecg waveform generator. *http://www.physionet.org/physiotools/ecgsyn*,[30 Apr 2009], 2003.

[17] Albacete Research Institute of Informatics. *http://www.i3a.uclm.es*.

[18] F. Mueller. Pthreads Library Interface. Institut fur Informatik, March 1999.

[19] T. Wagner and D. Towsley. Getting started with POSIX threads. Department of Computer Science, University of Massachusetts, July 1995.

[20] L. Dagum OpenMP: A Proposed Industry Standard API for Shared Memory Programming. OpenMP.org., 1997.

[21] L. Dagum and R. Menon. OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 46–55,1998.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)