CACIC 2003. IX CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACION

Implementación en paralelo de un modelo de transmisión sináptica sobre un cluster heterogéneo

Nombre de los autores, afiliación y direcciones

Diego Persano, Universidad Nacional de General Sarmiento (UNGS), Instituto de Ciencias, Informática. Campus Universitario, José M. Gutiérrez 1613 - Los Polvorines - Pcia. de Bs. As. - Argentina. Módulo 2, oficina 20. dpersano@ungs.edu.ar.

Adriana Angélica Gaudiani, Universidad Nacional de General Sarmiento (UNGS), Instituto de Ciencias, Informática. Campus Universitario, José M. Gutiérrez 1613 - Los Polvorines - Pcia. de Bs. As. - Argentina. Módulo 2, oficina 20. agaudi@ungs.edu.ar.

Dolores Isabel Rexachs, Universidad Autónoma de Barcelona (UAB), Arquitectura de Ordenadores y Sistemas Operativos, Dep. Informática, Edifici Q, 08193 Bellatera (Cerdanyola del Vallès), España, Dolores.Rexachs@uab.es

Emilio Luque, Universidad Autónoma de Barcelona (UAB), Arquitectura de Ordenadores y Sistemas Operativos, Dep. Informática, Edifici Q, 08193 Bellatera (Cerdanyola del Vallès), España, Emilio.Luque@uab.es

Abstract

En este trabajo se presentan los primeros resultados obtenidos por el área de informática de la Univ. Nac. de Gral. Sarmiento, en su proyecto de diseño e implementación de un cluster Linux, de tipo heterogéneo. Estos resultados se aplicaron en la implementación en paralelo de una aplicación desarrollada para el proyecto de investigación de Sistemas Complejos del área de física de dicha Institución. Como resultado de esta primera etapa se midió la capacidad de cómputo del cluster, obteniendo coeficientes que dependen de las características físicas de cada nodo y que permiten implementar aplicaciones paralelas, de manera estática, aprovechando al máximo el sistema de cómputo. El tiempo de ejecución de esta aplicación es de unas 20 hs. en su versión serial más simple y en las computadoras más rápidas, tiempo que irá creciendo en futuras aplicaciones. Su implementación en paralelo, utilizando dichos coeficientes, permitió disminuir un 60% el tiempo de ejecución. Mediante una implementación en paralelo, con asignación dinámica de la carga, se estudió el comportamiento del sistema al momento de balancear cómputo y comunicaciones, pensando continuar en el futuro mejorando este comportamiento e integrando este cluster a una colección de clusters heterogéneos interconectados a través de Internet.

Palabras claves:
Cluster, HNOW, pasaje de mensajes (MPI), Master/Worker, throughput, runtime.

Tema del Congreso:
Procesamiento Paralelo.

Implementación en paralelo de un modelo de transmisión sináptica sobre un cluster heterogéneo

Diego Persano[1], Adriana A. Gaudiani[1], D. Rexachs[2], E. Luque[2]

[1] *Area de Informática, Instituto de Ciencias, Universidad Naional. de General Sarmiento, Buenos Aires, Argentina*
dpersano@ungs.edu.ar, agaudi@ungs.edu.ar
[2] *Arquitectura de Ordenadores y Sistemas Operativos, Dep.Informática, Universidad Autónoma de Barcelona, España*
Dolores.Rexachs@uab.es, Emilio.Luque@uab.es

1. Introducción

Desde los inicios de la computación la demanda de mayor poder de cómputo se fue incrementando día a día y quedó claro que la solución no la brindaría la computación secuencial.
En muchas ramas de las ciencias la complejidad de los problemas que se estudian provocó la insistente exigencia de aumento del rendimiento mediante la reducción del tiempo de cómputo, surgiendo también la necesidad de contar con sistemas de procesamiento a tiempo real con respuestas más rápidas. Estos requerimientos impulsaron el procesamiento en paralelo. [1]
Al comienzo solo se podía contar con el acceso a una supercomputadora, máquinas poderosas que pueden desarrollar varios miles de millones de operaciones de punto flotante por segundo y que cuestan decenas de millones de dólares - precios que van más allá de los presupuestos de inversión de los grupos de investigación. La accesibilidad en el mercado a computadoras personales a bajo costo impulsó la iniciativa de comunicar varias de ellas para obtener una computadora poderosa a costos mínimos.
En los últimos años, el personal académico de diversas universidades y centros de investigación se han dado a la tarea de aprender a construir sus propias supercomputadoras conectando computadoras personales y desarrollando software para enfrentar tales problemas extraordinarios construyendo para ello los sistemas conocidos como clusters de PCs. En este tipo de sistema paralelo, cada CPU colabora con la otra realizando la tarea que le es asignada de manera individual, logrando entre todas funcionar como una única CPU más rápida.
Los constantes cambios en la tecnología de los componentes de estas computadoras y aprovechar el hardware existente en cada Institución hacen que estos clusters de PC´s estén integrados por máquinas heterogéneas, con diferentes velocidades de cálculos y diferentes tamaños de memoria principal y cache, como en el caso del cluster utilizado para este trabajo.
Al comienzo de nuestro trabajo analizamos el comportamiento de este cluster heterogéneo, mediante la implementación de un benchmark, con el objetivo de caracterizar su comportamiento y determinar coeficientes que relacionen el poder de cómputo de cada nodo que lo integra con la carga de trabajo a asignar a cada uno.
Utilizamos estos resultados para lograr un adecuado balanceo de carga a la hora de implementar aplicaciones paralelas. Primeramente utilizamos una asignación estática de la carga aprovechando los coeficientes hallados en la primera etapa, y posteriormente comparamos estos resultados con un reparto dinámico de la carga a cada nodo mediante una implementación master/worker del modelo de transmisión sináptica.
El modelo estático no permite independizarse de las características de cómputo de cada computadora del cluster, por lo cual dichos índices son imprescindibles para optimizar el uso del

sistema limitando en lo posible el tiempo ocioso de cada nodo. En cambio el modelo dinámico permite optimizar la distribución de la carga de trabajo entre los nodos del sistema logrando independizarse de la heterogeneidad, aunque la granularidad aplicada debe ser estudiada con cuidado para lograr aprovechar al máximo el sistema sin sobrecargar las comunicaciones. Este modelo evita que la máquina más lenta se convierta en el cuello de botella. [1] [2]
En la seccion 2 detallamos las características del sistema de cómputo que se utilizara. En la sección 3 mostramos los resultados de implementar el algoritmo benchmark y el cálculo de los coeficientes que brindan una medida de la heterogeneidad de la capacidad de cómputo del sistema.
En la sección 4 brindamos una descripción del modelo de transmisión sináptica para mostrar en la sección 5 los resultados obtenidos por su implementación en paralelo mediante un modelo estático y otro dinámico.
En la sección 6 presentamos las conclusiones y los trabajos futuros que complementarán esta presentación.

2. Caracterización del entorno

A continuación se presentan los distintos niveles que modelizan el sistema de cómputo sobre el cual se implementó la aplicación.
El nivel más bajo el sistema está constituido por un conjunto de computadoras de distinto poder de cómputo unidas por una red de conexión, trabajando bajo Linux, que llamaremos Cluster heterogéneo o HNOW.
El cluster está compuesto por cuatro computadoras interconectadas con placas Ethernet de 100 Megabits. Una de las computadoras, Pegasus, posee dos placas Ethernet. Una de ellas lo conecta a la red de la Institución y a Internet, la otra a la LAN local formada por las otras máquinas del cluster. Las computadoras trabajan bajo sistema operativo Gnu/Linux, distribuciones RedHat y Debian, y tienen instalado el sistema de distribución de archivos compartido NFS, siendo Pegasus la computadora servidora de archivos. [3]

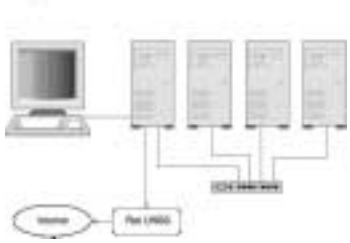

Figura 1

El nivel siguiente es el que permite la comunicación entre los nodos. Para implementar programas en paralelo sobre este hardware se necesitan lenguajes que permitan utilizar el sistema como una única máquina virtual ofreciendo mecanismos de comunicación y de sincronización. La manera en que se comunican las computadoras que integran el cluster es mediante el pasaje de mensajes utilizando las librerías MPI. [4]
El nivel superior del sistema es el de implementación de la aplicación, y requiere seleccionar el paradigma adecuado a las características del entorno sobre el que trabajamos, que en nuestro caso es un cluster heterogéneo, y adecuado a las características propias del algoritmo.
Primeramente implementamos el algoritmo benchmark en paralelo con distribución estática de la carga utilizando los índices de heterogeneidad de cada máquina del cluster calculados en la primera etapa de las experiencias, posteriormente comparamos este rendimiento con el obtenido al implementar el modelo de transmisión sináptica de manera estática y mediante un modelo jerárquico Master/Worker que permitió una distribución dinámica de la carga.

3. Benchmark

En una primera etapa determinamos las características del hardware de cada una de las computadoras que componen el cluster, (La Tabla 1 muestra estos datos) y medimos la capacidad de cómputo de los nodos que lo componen.

Tabla 1. Caracterización de los nodos del Cluster

Nodo	CPU	Memoria	Cache
Pegasus	Pentium III – 866 Mhz	512 MB	32 KB
Orion	Pentium III - 526 Mhz	512 MB	32 KB
Andrómeda	AMD – 500 Mhz	128 MB	64 KB
Centauro	AMD – 250 Mhz	128 MB	64 KB

Para medir la capacidad de cómputo del sistema realizamos mediciones de tiempo de ejecución de cada máquina utilizando como benchmark el algoritmo de multiplicación de matrices MMM – Matrix Matrix Multiplication. Este algoritmo constituye un importante algoritmo del álgebra lineal que se caracteriza por ser altamente escalable y fácilmente modificable.[5][6]
Estas mediciones se realizaron mediante la implementación de un algoritmo serial que corrió individualmente en cada nodo y para matrices de distintas dimensiones. Los resultados obtenidos fueron los tiempos de ejecución en segundos (Figura 2) y en megaflops (Figura 3) para cada nodo.

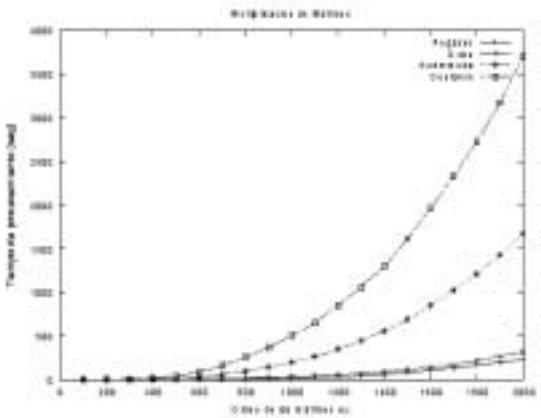

Figura 2

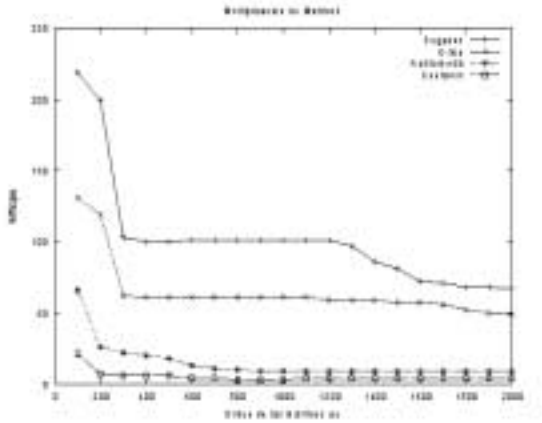

Figura 3

Las características de la memoria principal y la memoria cache de cada computadora influyen en el comportamiento de las curvas de megaflops en función del tamaño de las matrices haciendo que se mantenga casi constante en un amplio rango de n, luego de haber disminuido bruscamente para dimensiones menores a 300. Los valores alcanzados en megaflops son similares para los pares de nodos Pegasus-Orión y Andrómeda-Centauro, los cuales presentan entre sí características similares en sus niveles de memoria.
En una segunda etapa buscamos una medida para la distribución de cargas entre los nodos del cluster en función de su heterogeneidad. Como el algoritmo MMM es de complejidad $O(n^3)$, requiere tiempo de cómputo muy grande para matrices grandes, por lo cual constituye un benchmark interesante al momento de paralelizarlo, mediante una adecuada carga de datos. [5]
Con estos resultados pudimos medir el poder de cómputo de cada nodo del cluster. Usamos esta información para optimizar el runtime cuando todos los procesadores cooperan en el cómputo en paralelo.
El método que utilizamos fue enviar la matriz A a todos los nodos y enviar filas de la matriz B de modo proporcional al poder de cómputo de cada uno.
Una medida de la capacidad de proceso del sistema lo da el Grado de Heterogeneidad del mismo, que en este cluster se midió corriendo el benchmark para matrices de 1000x1000.

Tabla 2. Grado de Heterogeneidad

Tiempo de Ejecución		Grado de Heterogeneidad
Máquina más rápida	Máquina más lenta	
19550 msg.	51200 msg.	0,38

Sabiendo que el nodo más rápido tarda un tiempo t en procesar las n filas de B, se calculó la cantidad de filas que se debía enviar a cada uno de los otros nodos para que emplee el mismo t en procesarlas. Luego de intentar con valores de n entre 100 y 2000, se promediaron los tiempos medidos y se obtuvo un coeficiente en función de n que indica la cantidad de filas a distribuir entre cada nodo. Estos valores son:

Tabla 3. Coeficientes de cada nodo

Pegasus	Orión	Andrómeda	Centauro
0,5	0,36	0,10	0,04

En la figura 4 se pueden ver los resultados obtenidos con la paralelización del algoritmo MMM en el modo estático, distribuyendo la carga según los coeficientes anteriores.

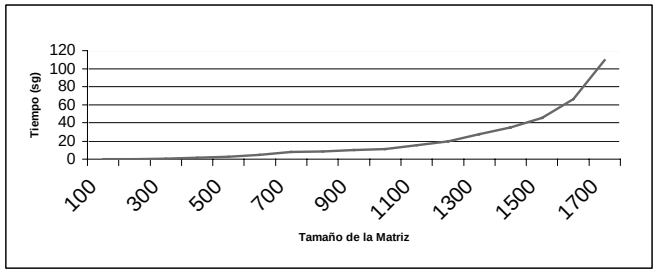

Figura 4

Este método disminuyó en un promedio del 30% el tiempo total de ejecución del algoritmo MMM comparándolo con el mejor tiempo logrado en modo serial.

4. Descripción del modelo a implementar

El problema de la transmisión sináptica es una modelización de fenómenos físicos que, como muchas de ellas, utilizan la computadora como laboratorio natural para la validación de las teorías resultantes. Debido a que en estos procesos intervienen muchos grados de libertad, son problemas que día a día incrementan su intensidad de cómputo. Por ello, la necesidad de acelerar los tiempos de cómputo demanda arquitecturas que ofrezcan más potencia computacional, encontrando solución en los sistemas de cómputo paralelo, especialmente en clusters de PCs.
Mediante simulación numérica se estudia la propagación de una señal periódica de baja frecuencia a través de una red de osciladores biestables acoplados unidireccionalmente, operando en un régimen sobreamortiguado y sujetos a la suma de ruido blanco gaussiano. Se asume que los osciladores operan bajo un régimen de resonancia estocástica. Este fenómeno es de importancia, en neurofisiología, en la transmisión sináptica entre neuronas, las que son modeladas por sistemas biestables con dos configuraciones estables correspondientes a los estado de “inactivo” y de “disparo” del axón. Este trabajo es la continuación de estudios previos sobre la transmisión

sináptica entre neuronas, simuladas mediante cadenas y anillos de osciladores armónicos.[7] [8] [9] Cuando los parámetros del sistema biestable están apropiadamente sintonizadas, la interacción entre el ruido externo y el mecanismo de conducción de la señal produce un pico en el espectro de potencia del sistema que corresponde al máximo de la relación señal-ruido. Este pico muestra la difusión de la potencia de la señal, a través de todo el espectro de ruido, de un modo coherente con la entrada [8]
La simulación numérica permite estudiar el comportamiento de la transmisión cuando se interrumpe accidentalmente la red, por ejemplo si un oscilador es privado de ruido, demostrando la robustez del sistema ante esta situación. Este es un aspecto importante del acople de neuronas: la continuidad de la transmisión del pico de potencia de una neurona "muerta" en la red. [8] [9]
Con este modelo se estudia el comportamiento de la transmisión de la señal a través de la red cuando alguno de los osciladores es privado de ruido. Se encuentra un valor de acople entre los osciladores tal que por debajo de este la propagación puede ser considerada interrumpida en la "neurona muerta" y es restablecida por encima de este. Este comportamiento se le llama tolerante a fallos e incrementa la confiabilidad a expensas de la eficiencia.

5. Implementación del modelo de transmisión sináptica

Este algoritmo realiza una gran cantidad de cómputo, ya que debe promediar los valores resultantes de la integración, que se implementa con al menos 2^{17} pasos de iteración, repitiendo este cálculo para una cantidad no menor de 100 juegos de condiciones iniciales.
A continuación se describe el algoritmo serial:

1) Se establecen los parámetros del problema como en McNamara-Wiesenfeld [7]
2) Se repite *Numproc* (>100) veces el proceso total para promediar y disminuir el ruido.
   i) Se eligen las condiciones iniciales de cada oscilador de la red de dimensión *line* x *col* (30 x 30) utilizando las funciones Random y Gauss de Numerical Recipes.
   ii) Se realizan *tfinal* (2^{17} - o en general 2^n) pasos de integración.
      (1) En cada uno de ellos se simula la propagación de la señal en la red.
      (2) De cada uno de los 30 valores inciales de la matriz, se toman todos los obtenidos en los *tfinal* pasos de integración y se les aplica la Transformada de Fourier, utilizando el algoritmo FFT de Numerical Recipes
      (3) Se guardan los primeros *Nmaximo* (600) valores resultantes del proceso anterior para cada uno de los osciladores (30x30) en un arreglo de potencias.
3) Se promedian los valores del espectro de potencia obtenidos para las *Numproc* ejecuciones.

Una característica de este algoritmo, que inicialmente fue escrito en lenguaje Fortran, es la demanda de gran cantidad de prestaciones de nuestro equipo, CPU y memoria. Por esta razón fue traducido a lenguaje C para mejorar el uso de la memoria mediante el uso de memoria dinámica.
Este algoritmo, a diferencia del MMM, pertenece a un tipo de problemas que no se distinguen por el paralelismo de datos. El objetivo de la paralelización fue aumentar el rendimiento general del sistema o throughput, o sea la cantidad de tareas que realiza el sistema de cómputo por unidad de tiempo. De esta forma se logra procesar en tiempos que son más satisfactorios para el usuario de la aplicación.
Inicialmente implementamos una versión en paralelo de este algoritmo utilizando un modelo estático de reparto de cargas y como referencia, para medir la mejora del tiempo de ejecución en paralelo, medimos el tiempo de ejecución en serie en la máquina más potente del cluster, Pegasus, siendo el tiempo total de 1.230 minutos (20,5 hs.)

Utilizando los coeficientes que se muestran en la Tabla 3, repartimos las tareas a los cuatro nodos del cluster obteniendo un tiempo total de ejecución de 648 minutos. (Casi 11 hs.). Este resultado mostró un 52,7% de reducción del tiempo de ejecución serial, corriendo la aplicación con 200 juegos de condiciones iniciales.

En una segunda etapa ejecutamos el modelo mediante una implementación dinámica master-worker, logrando alcanzar el mejor tiempo total de ejecución, de 580 min, cuando cada worker procesaba 25 juegos de condiciones iniciales. Para obtener estos resultados utilizamos como Master la computadora con menor poder de cómputo, Centauro. Se trabajó con distinta carga en los workers tratando de equilibrar el overhead de comunicaciones con el tiempo libre de cada nodo. (Tabla 4)

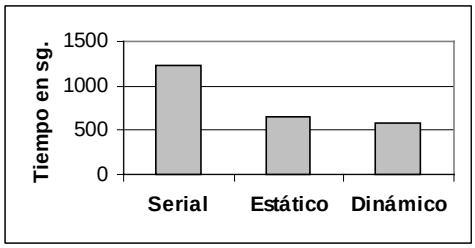

Figura 5

Tabla 4. Tiempo de ejecución total según la carga de trabajo que realiza cada worker

Carga de trabajo	5	10	20	25	40	50
Tiempo	790 m.	798 m.	598 m.	580 m.	890 m	1100 m.

Cuando la granularidad es pequeña se incrementan los tiempos de comunicaciones y cuando la granularidad aumenta es mayor el tiempo ocioso del nodo más rápido. Ambos casos aumentan el runtime.

Aunque la mejora en el rendimiento de la aplicación lograda al implementar el método estático fue suficiente para los intereses de sus usuarios, se implementó también el modelo dinámico logrando muy poca reducción en el tiempo de cómputo, comparándolo con el caso estático.

El algoritmo dinámico fue de interés en el área de informática ya que permitió medir el rendimiento general del cluster determinando el tiempo adicional en que las comunicaciones penalizan el tiempo total de ejecución. Se trató de encontrar una carga óptima de tareas que optimice el tiempo de ejecución minimizando el tiempo ocioso de cada nodo sin recargar el sistema con tiempo de comunicación entre master y workers.

6. Conclusiones

La implementación en paralelo de esta aplicación fue el primer intento de analizar el comportamiento de nuestro cluster heterogéneo y de optimizar su uso cuando los usuarios de este recurso corran sus aplicaciones paralelas, sabiendo que el único objetivo de dichos usuarios es disminuir el runtime de sus aplicaciones.

Para lograr esto es necesario que les brindemos un sistema de cómputo paralelo con sus recursos optimizados. Pero, también se necesita alguna medida del sistema que les permita realizar las implementaciones en paralelo de manera eficiente. El uso de los coeficientes calculados brinda una medida de la colaboración de cada nodo al momento de correr una aplicación paralela estática que utilice a todos ellos a la vez.

La continuación de este trabajo tiene como propósito la integración de este cluster a una colección de clusters de características similares (CoHNOW) participando de las investigaciones que está realizando en esta línea la Unidad de Arquitectura y Sistemas Operativos de la Universidad Autónoma de Barcelona, España, en conjunto con el Curso de Informática de la Universidad Católica de Salvador, Bahía, Brasil.[2]

El grupo de investigación que participa de este proyecto ha avanzado tratando de minimizar el costo que implica utilizar una red inestable e impredecible de interconexión entre los clusters, como es Internet, lo cual no es un problema trivial.
Utilizando un modelo master-worker, en el que se tiene un cluster (HNOW) que funciona como Master y los restantes sub-clusters funcionan como Workers, se lograría la colaboración entre los tres clusters (CoHNOW) para el cómputo paralelo, de manera que el sistema funcione como una máquina virtual siendo transparente esta arquitectura para el usuario que corra aplicaciones a gran escala.

Agradecimientos:

Agradecemos el financiamiento parcial del CONICET, a través del proyecto PIP 4210, para la construcción del cluster de la UNGS.

7. Bibliografía

[1] Souza J.R., *Influencia de la comunicación en el rendimiento de un sistema de computación paralela, basado en redes de estaciones de trabajo,* Tese de Mestrado, Arquitetura de Computadores e Processamento Paralelo, Universidade Autônoma de Barcelona, UAB, 2000.
[2] Furtado, Adhvan. Souza, Josemar, Rebouças, André. Rexachs, Dolores. Luque, Emilio. Architectures for an Efficient Application Execution in a Collection of HNOWS. In: D. Kranzlmüller et al. (Eds.):Euro PVM/MPI 2002, LNCS 2474,pp.450-460,2002.
[3] Sterling, Thomas L. Salmon, John. Becker, Donald J. e Savaresse, Daniel F. *How to build a Beowulf: a guide to the implementation and aplication of PC clusters.* Massachusetts Institute of Technology. 1999
[4] W. Gropp, E. Lusk, R. Thakur, *Using MPI-2: Advanced Features of the Message-Passing Interface,* Scientific and Engineering Computation Series, Massachusetts Intitute of Technology, 1999.
[5] Tinetti, A. Quijano, A. Giusti, E. Luque, "Heterogeneous Network of Workstations and the Parallel Matrix Multiplication", *Euro PVM/MPI 2001*, Y. Cotronis and J. Dongarra, eds., pp. 296-303, 2001.
[6] Beaumont, F. Rastello and Y. Robert. "Matrix Multiplication on Heterogeneous Platforms", *IEEE Trans. On Parallel and Distributed Systems*, vol. 12, No. 10, October 2001.
[7] McNamara, K. Wiesenfeld, "Theory of Stochastic Resonance". Phys. Rev. A 39 (1989) 4854.
[8] Perazzo R., Romanelli, Deza, "Fault tolerance in noise-enhanced propagation". Phys.Rev. E 61 (2000) R 3287.
[9] M.F. Carusela, R. Perazzo, L.Romanelli, "Information transmisión and storage sustained by noise". Elsevier Science B.V. Physica D. (2002) 177-183