STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **L**, Number 1, 2005

PARALLEL LAGRANGE INTERPOLATION ON EXTENDED FIBONACCI CUBES

IOANA ZELINA

Abstract. In this paper is presented a parallel algorithm for computing a Lagrange interpolation on a Extended Fibonacci Cube $EFC_1(n)$.The algorithm consists of three phases: initialisation phase, main phase in wich the Lagrange polynomials are computed and final phase in wich the terms of the interpolation formula are added together.

1. Introduction

Interpolation techniques are of great importance in numerical analysis since they are used in many science and engineering domains. The Lagrange interpolation for a given set of points $(x_1, y_1), (x_2, y_2)$,..., (x_N, y_N) and a value x is defined as

$$f(x) = \sum_{i=1}^{N} y_i \times L_i(x) \tag{1.1}$$

where $L_i, i = \overline{1, N}$ are the Lagrange polynomials given by the formula

$$L_i(x) = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_N)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_N)} \tag{1.2}$$

When the number of points N is very large a long computation time and a large storage capacity may be required to carry out the calculation. To overcome this, a parallel implementation would be appropriate. This kind of parallel algorithms were introduced for Lagrange interpolation for different topologies: Goertzel [2] has introduced a parallel algorithm for a tree topology with N processors augmented with ring connections which requires $N/2 + O(\log N)$ steps each composed of two substractions and four multiplications; a parallel algorithm has been discussed in [6] which uses a k-ary n-cube consisting of $O(k^n + kn)$ steps, each with 4 multiplications and substractions for $N = k^n$ node interpolation. In [5] is described a parallel algorithm for computing a $N = n!$-node Lagrange interpolation on a n-star graph. The algorithm in [5] consists of three phases and requires $n!/2$ steps, each consisting of 4 multiplications, 4 substractions and one communication operation. In [7] this parallel algorithm is applied for computing an $N = n2^n$ point Lagrange interpolation on an n-dimensional cube-connected cycles (CCC_n).

Received by the editors: February 18, 2005.

The method can be applied for any Hamiltonian network, the performances depending on the communication abilities of the host network.

In this paper the algorithm described in [5] is applied to a Extended Fibonacci Cube topology. The algorithm relies on all-to-all broadcast communication at some stages during computation. This is achieved by using a gossiping algorithm on a ring embedded in the host network having its all nodes.

2. Preliminaries

Due to the regularity, logarithmic diameter, logarithmic node degrees, etc. of hypercube interconnection networks, they are used by most researchers. The hypercube provides a rich interconnection structure which permits many other topology to be emulated. Nevertheless, when dimension of hypercube increases, the number of nodes increases too fast. Because of this, Hsu [3] developed Fibonacci Cubes and they are more sparse than the hypercubes and Wu [8] developed Extended Fibonacci Cubes by changing the initial conditions. The Extended Fibonacci Cubes are also more sparse than the hypercubes.

One third of Fibonacci Cubes are Hamiltonian, however, all of Extended Fibonacci Cubes are Hamiltonian. Both of these interconnection can be considered as nodes faulty with incident edges hypercubes.

Fibonacci Cubes and Extended Fibonacci Cubes

Extended Fibonacci Cubes (EFC) topology was proposed by Wu [1] and this topology is based on the Fibonacci Cube proposed by Hsu [3]. Both topologies use the Fibonacci series and initial conditions for topologies can be different from Fibonacci series initial conditions.

An k^{th} $(k = 1, 2)$ order Extended Fibonacci Cube is denoted by $EFC_k(n)$ where $n-2$ is the length of bitstring representing the address of nodes in EFC. $EFC_k(n)$ is a subgraph of the corresponding hypercube. Each node of $EFC_k(n)$ is addressed with Fibonacci Code (FC). The simplest version of these cubes series is the Fibonacci Cubes. The Fibonacci Cube $(FC(n))$ can be described as below.

Definition 2.1. [Fibonacci Cube] *Assume the graphs $FC(n) = (V(n), E(n))$, $FC(n-1) = (V(n-1), E(n-1))$ and $FC(n-2) = (V(n-2), E(n-2))$. We define the Fibonacci Cube using the recursion for the nodes set as $V(n) = 0\|V(n-1) \cup 10\|V(n-2)$, where $\|$ denotes the concatenation of two bit-strings. Two nodes in $FC(n)$ are connected by an edge in $E(n)$ if and only if their labels differ exactly in 1-bit position. The initial condition for recursion is $V(2) = \emptyset$ and $V(3) = \{0, 1\}$.*

Definition 2.2. [Extended Fibonacci Cube] *Let $EFC_1(n) = (V_1(n), E_1(n))$, where $V_1(n)$ is the set of nodes and $E_1(n)$ is the set of edges in $EFC_1(n)$, and*

$$EFC_1(n-1) = (V_1(n-1), E_1(n-1)), EFC_1(n-2) = (V_1(n-2), E_1(n-2)).$$

$EFC_1(n)$ can be defined recursively by using $EFC_1(n-1)$ and $EFC_1(n-2)$ as it follows: $V_1(n) = 0\|V_1(n-1) \cup 10\|V_1(n-2)$ where $\|$ denotes the concatenation of two strings. The initial condition for recursion is $V_1(3) = \{0, 1\}$ and $V_1(4) = \{00, 10, 11, 01\}$. Two nodes in $EFC_1(n)$ are connected if and only if their address representations differ in exactly 1-bit position.

Some $EFC_1(n)$ are shown in Fig.1 where $n = 3, 4, 5, 6$ and each $EFC_1(n)$ consists of an $EFC_1(n-1)$ and an $EFC_1(n-2)$.

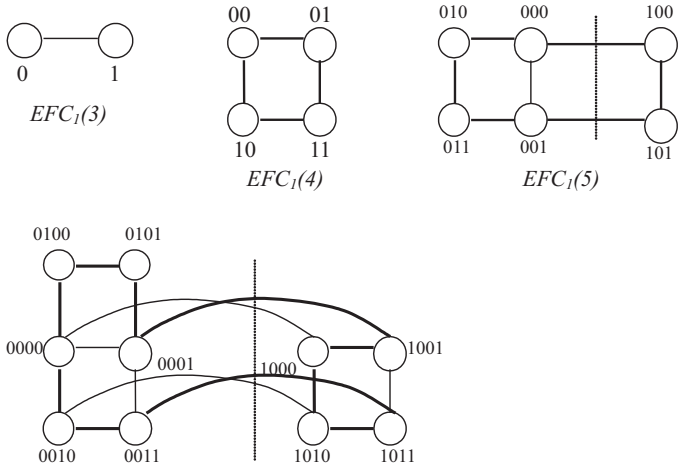

Fig. 1

It is known that while less than one third of Fibonacci cubes are hamiltonian, all of $EFC_k(n)$ are Hamiltonian. This can be proved using inductive reasoning on n, where $n = 4$ and $n = 5$ are the induction basis. In Fig. 1 a hamiltonian cycle is shown with bold links. This means that ring can be embedded into $EFC_k(n)$ with dilation and congestion 1. $FC(n)$ is a proper subgraph of $EFC_1(n)$.

There is a Hamming distance path in $EFC_1(n)$ where Hamming distance is the exclusive-or operation on both addresses of nodes and this distance is equal to Hamming distance. The diameter of $EFC_1(n)$ is $n-2$ and node degrees are between $\lceil \frac{n}{3} \rceil$ and $n-2$.

By changing the initial conditions for $EFC_1(n)$, another Extended Fibonnaci Cube can be extracted, denoted as $EFC_2(n)$ with initial conditions $V_2(4) = \{00, 10, 11, 01\}$ and $V_2(5) = \{000, 100, 101, 111, 110, 010, 011, 001\}$.

$EFC_1(n)$ is a proper subgraph of $EFC_2(n)$. The generated FC for $EFC_1(n)$ and $EFC_2(n)$ are mutually disjoint.

We denote by N the number of nodes in $EFC_1(n)$.

3. The Parallel Algorithm

The parallel algorithm is based on the algorithm described in [5] for computing a $N = n!$ node Lagrange interpolation on a n-star graph. We shall apply this algorithm for a network using an n- extended Fibonacci cube $EFC_1(n)$ topology with bidirectional links between nodes. Let N be the number of the nodes in $EFC_1(n)$.

The computation is carried out in three phases: initialisation, main and final phase. In the initialisation phase, the set of points to be interpolated are allocated to the nodes, one point for each node. Then, in the main phase, the Lagrange polynomials $L_i(x)$,$i = \overline{1,N}$ are computed and in the final phase the terms are added together to obtain the final result $y = f(x)$.

We denote by P_w the processor in the node of the extended Fibonacci cube $EFC_1(n)$ with the binary representation w. Each processor P_w has six registers denoted R_1, R_2, R_3, R_4, R_5, R_6 and we indicate by $P_w(R_i)$ the content of the register R_i in the processor P_w, $i = \overline{1,6}$, $w \in V_1(n)$ and by $P_w^{(t)}(R_i)$, $i = \overline{1,6}$, $w \in V_1(n)$ the content of the register R_i in the processor P_w after step t. In each node, registers R_1, R_2, R_3, R_4 will hold the terms required for computing the polynomials and registers R_5, R_6 will be used to implement an all-to-all broadcast algorithm in a ring embedded in the host network $EFC_1(n)$ during the main phase.

The $EFC_1(n) = (V(n), E(n))$ is hamiltonian. When constructing a hamiltonian cycle in $EFC_1(n)$, two arrays, $Next[w]$ and $Previous[w]$, which indicate the nodes before, respectively after node w, $w \in V(n)$ in the embedded cycle can also be constructed. For any node P_w in the embedded hamiltonian ring, the next and previous nodes are $P_{Next[w]}$, respectively $P_{Previous[w]}$. Those arrays should have been set to their proper values before starting the initialisation phase.

3.1. **Initialisation phase.** The values x, $Next[w]$, $Previous[w]$, (x_i, y_i) are assignated to the processor P_w to be stored in the local memory where i is the order of the Fibonacci number $w = f_i$, $i = \overline{1,N}$ with initial conditions from Definition 2.2. The registers R_1, R_2, R_3, R_4, R_5, R_6 of each processor are set to their initial values, for all $w \in V(n)$, in parallel:

$$P_w^{(0)}(R_1) = 1; \quad P_w^{(0)}(R_2) = 1; \quad P_w^{(0)}(R_3) = x_i; \quad P_w^{(0)}(R_1) = x_i;$$
$$P_w^{(0)}(R_5) = x - x_i; \quad P_w^{(0)}(R_6) = x - x_i;$$

3.2. **Main phase.** In this phase, each node P_w uses the values $Next[w]$ and $Previous[w]$ to communicate with the next and previous node in the embedded hamiltonian cycle. To compute the terms $L_i(x)$ all the processors perform the following sequence simultaneously:

For $t = 0, 1, ..., N/2 - 2$ do

$$P_w^{(t+1)}(R_3) \Leftarrow P_{Next[w]}^{(t)}(R_3);$$
$$P_w^{(t+1)}(R_4) \Leftarrow P_{\mathrm{Pr}evious[w]}^{(t)}(R_4);$$
$$P_w^{(t+1)}(R_5) \Leftarrow P_{Next[w]}^{(t)}(R_5);$$
$$P_w^{(t+1)}(R_6) \Leftarrow P_{\mathrm{Pr}evious[w]}^{(t)}(R_6);$$
$$P_w^{(t+1)}(R_1) = P_w^{(t+1)}(R_1) \times P_w^{(t+1)}(R_5) \times P_w^{(t+1)}(R_6);$$
$$P_w^{(t+1)}(R_2) = P_w^{(t+1)}(R_2) \times (x_i - P_w^{(t+1)}(R_3)) \times (x_i - P_w^{(t+1)}(R_4));$$

end for;

$$P_w^{(N/2)}(R_3) \Leftarrow P_{Next[w]}^{(N/2-1)}(R_3);$$
$$P_w^{(N/2)}(R_1) = P_w^{(N/2)}(R_1) \times P_w^{(N/2)}(R_5);$$
$$P_w^{(N/2)}(R_2) = P_w^{(N/2)}(R_2) \times (x_i - P_w^{(N/2)}(R_3));$$

The last iteration is used to avoid multiplying the terms $(x-x_{N/2})$ and $(x_i-x_{N/2})$ twice.

Each step consists of two data communications (the first two respectively the last two communications can be realized in parallel because of bidirectional links between nodes), 2 substractions and 4 multiplications.

To conclude the main phase, all the processors execute the instruction

$$P_w^{(N/2+1)}(R_1) = \frac{P_w^{(N/2)}(R_1)}{P_w^{(N/2)}(R_2)} \times y_i.$$

Therefore, at the end of this phase $P_w(R_1) = L_i(x) \times y_i$.

In the main phase, each processor performs N data communications, $2N-1$ multiplications, $N-1$ substractions and one division.

3.3. **The Final Phase.** In this phase, the contents of register R_1 in all nodes are added together to obtain the final result. We can use for this a gossiping method for a ring similar to the one used in the main phase but we can also use a method similar to the addition of the content of the processors in a hypercube network topology.

Remark that if a node $w \in V(n)$ in $EFC_1(n)$ is labeled with a bit string having 1 on his first position, $w = 1u_2u_3 \dots u_{n-3}$ then $u_2 = 0$ and the node $w' = u_3 \dots u_{n-3} \in V(n-2)$ is a node in $EFC_1(n-2)$. A node $w \in V(n)$ in $EFC_1(n)$ is labeled with a bit string having 0 on his first position, $w = 0u_2u_3 \dots u_{n-3}$ then the node $w' = u_2u_3 \dots u_{n-3} \in V(n-1)$ is a node in $EFC_1(n-1)$. If we add simultaneously the content of registers R_1 in all the nodes $w = 1u_2u_3 \dots u_{n-3}$ to the content of registers R_1 in the corresponding nodes $v = 0u_2u_3 \dots u_{n-3}$ then we reduce the size of the EFC from n to $n-1$. So we can add together the partial results accumulated in registers R_1 of all nodes in n steps, each consisting of one addition and two communication operations. Each step reduces the size of problem by one until the last step, $(n-2)^{th}$ step, which complete the computation having stored the final result in register R_1 of processor $P_{0\dots0}$.

For $i = 1, 2, ..., n-2$ do
  For all $w = 0^{i-1}1u_{i+1} \dots u_{n-2}$ do in parallel

$$P_{0^i u_{i+1} \dots u_{n-2}}(R_1) = P_{0^{i-1}1 u_{i+1} \dots u_{n-2}}(R_1) + P_{0^i u_{i+1} \dots u_{n-2}}(R_1);$$

  end for;

End for;

This phase includes $n-2$ additions and $n-2$ communication operations.

Let n be the order of the Extended Fibonacci Cube $EFC_1(n)$ and N be the dimension of the $EFC_1(n)$, i.e. the extended Fibonacci number defined by the recursion $f_n = f_{n-1} + f_{n-2}$, $n \geq 2$, $f_0 = 2$, $f_1 = 4$.

Theorem 3.1. *The parallel algorithm presented carries out the computation of a N-point Lagrange interpolation on a Extended Fibonacci Cube $EFC_1(n)$ in a total time of $O(N)$.*

Proof: The algorithm computes a N-point interpolation, in three phases, requiring in total $N+n-2$ data communications, $n-2$ additions, $2N-1$ multiplications, $N-1$ substractions and one division.

Remark 3.1. *The parallel algorithm carries out in a total time of $O(N)$ while the running time for such an interpolation on a single-processor system is of $O(N^2)$.*

References

[1] Akl, S., *Parallel Computation: Models and Methods*, Prentice Hall, 1997
[2] Goertzel, B., *Lagrange interpolation on a tree of processors with ring connections*, JPDC, 22, pp.321-333, 1994
[3] Hsu, W.J., *Fibonacci Cube- A New Interconnection Topology*, IEEE Trans. on Parallel and Distributed Systems, vol.4, no.1, pp.3-12, 1993
[4] Karci, A., *New Interconnection Networks: Fibonacci Cube and Extended Fibonacci Cubes Based Hierarchic Networks*, Proc. of 15th ICOIN, 2001
[5] Sarbazi-Azad, H., Ould-Khaoma, M., Mackenzie, L.M., *A Parallel Lagrange Interpolation on the Star Graph*, Proc. 14th IPDPS, Cancun, Mexico, pp.777, 2000
[6] Sarbazi-Azad, H., Ould-Khaoma, M., Mackenzie, L.M., *A Parallel Lagrange Interpolation on k-ary n-cubes*, LNCS1557, pp.85-95, 1999
[7] Sarbazi-Azad, H., Ould-Khaoma, M., Mackenzie, L.M., *An Efficient Parallel Algorithm for Lagrange Interpolation and Its Performance*, Proc. 4th Int.Conf. on High Performance Conputing in Asia Pacific Region, vol 2, Beijing, China, pp.593, 2000
[8] Wu, J., *Extended Fibonacci Cubes*, IEEE Trans. on Parallel and Distributed Systems, vol.8, no.12, pp.1203-1210, 1997

North University of Baia Mare, Faculty of Science, Department of Mathematics and Computer Science, Victoriei 76, Baia Mare, Romania
E-mail address: ioanazelina@yahoo.com