A Parallel Algorithm for Interpolation in Pancake Graph

IOANA ZELINA, PETRICA POP, CORINA POP SITAR, IOANA TASCU
Department of Mathematics and Computer Science
North University of Baia Mare
Victoriei 76, Baia Mare
ROMANIA

Abstract: In this paper a parallel algorithm for Lagrange interpolation is applied on a *n*-pancake graph. The *n*-pancake graph is a Cayley graph with $N=n!$ vertices and with attractive properties regarding degree, diameter, symmetry, embeddings and self similarity. Using these properties the algorithm carries the calculation in $O(N)$ steps of communication and arithmetic operations instead of $O(N^2)$ steps for a single processor system.

Key-Words: interconnection topology, pancake graph, Lagrange interpolation

1 Introduction

An interconnection network consists of a set of processors, each with a local memory and a set of bidirectional links that serve for the exchange of data between processors. A convenient representation of an interconnection network is by an undirected (or sometimes directed) graph $G=(V, E)$ where each processor P is a vertex in V and two vertices are connected by an edge in E if and only if there is a direct (bidirectional for undirected and unidirectional for directed graphs) communication link between processors. Such processors are called neighbors. The interconnection graph of a network is referred to as its topology. We will use node, vertex and processor with the same meaning and the terms edge and link are used as synonyms. Usually, all processors in a network are identical and each is assumed to have input and output abilities. Processors may execute the same or different programs. The time complexity of any algorithm has two components: computation time (covers local computation) and communication time (the time needed for the exchange of data between processors).

We can say that a network topology is "good" if it has some properties as: small degree for the nodes, small diameter (that means small delay in communication), maximum connectivity (good fault tolerance), symmetry (minimum congestion, uniform loading), embedding properties (good simulation of other networks), modular structure (offering the possibility of recursive decomposition).

The pancake topology used in this paper has a lot of such good properties that make this topology very attractive.

2 Problem Formulation

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

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

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

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

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 $N/2 + O(1\log N)$ steps each composed of two substractions and four multiplications; a parallel algorithm has been discussed in [7] 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 [6] is described a parallel algorithm for computing a $N = n!$-node Lagrange interpolation on a *n*-star graph. The algorithm in [6] consists of three phases and requires $n!/2$ steps, each consisting of 4 multiplications, 4 substractions and one communication operation. In [8], this

parallel algorithm is applied for computing an $N=n2^n$ point Lagrange interpolation on an *n*-dimensional cube-connected cycles (CCC_n) and in [10] the algorithm is applied for computing a Lagrange interpolation on an extended Fibonacci cube.

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 [6] is applied to a pancake 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.

2.1 The Pancake Graph

The pancake was proposed by Akers and Krishnamurthy [1] together with the star as alternatives to the hypercube for interconnecting processors in parallel computers. These networks have good properties: edge and vertex symmetry, small degree and diameter, extensibility, high connectivity (robustness), easy routings and broadcasting. Properties of the pancake graph were studied in [1], [3], [5], [9]. The model used to define and analyze the pancake (used for the star to) is a group-theoretic model:

Definition. Let *(G, •)* be a finite multiplicative group, *I* the identity in *G* and *S* a set of generators of *G* with the properties:

a) $\forall g \in S \quad g^{-1} \in S$;

b) $I \notin S$.

A ***Cayley graph*** *(V, E)* is defined as a graph whose vertex set is *V= G* and the edge set is $E = \{(u,v) \in V \times V \mid u^{-1}v \in S\}$.

The Cayley graphs are finite, connected, undirected, devoid of multiple edges, loop free and symmetric.

Definition. The ***pancake network*** *PC(n)* of dimension *n* is the Cayley graph $P_n = (S_n, E)$ whose set of generators is $S = \{g_i \in S_n \mid g_i = (i(i-1)...321(i+1)...n) \mid i = \overline{2,n}\}$ and S_n is the set of the permutations of the elements *{1, 2,..., n}*.

In other words, the *n!* vertices of a *PC(n)* are labeled with the permutations on *n* symbols and any two vertices of *PC(n)*, $u = x_1x_2...x_n$ and $v = y_1y_2...y_n$, are connected iff there exists an integer *i*, $2 \le i \le n$ such that $y_j = x_{i-j+1}$ for $j = \overline{1,n}$ and $y_j = x_j$ for $j > i$.

There are *(n-1)* generators, one for each value of *i*, $2 \le i \le n$ and $|S| = n-1$. The Cayley network *PC(n)* has *n!* vertices, each with degree $|S| = n-1$ and *PC(n)* is *(n-1)*- regular. It is clear that de node degree of *PC(n)* is of order *O(logN), N=n!*, sublogarithmic in the number of processors. The diameter of the pancake *PC(n)* is less than $\frac{5n+5}{3}$.

The pancake *PC(n)* can be decomposed into *n* subpancakes of dimension *n-1*. Each of the *(n-1)!* vertices of each *(n-1)*- subpancake has a block representation of the form *Ai* where $A \in S_{n-1}$ is a permutation block on the *(n-1)* symbols *{1, 2, ... ,i-1, i+1,..., n}* for a given $i \in \{1,...,n\}$ which depends on the considered subpancake.

Example. For *n=4*, the pancake *PC(4)* is the graph presented in Fig 1. In this case the set of generators is *S={(2134),(3214),(4321)}* and the pancake P_4 consist of 4 copies of *PC(3)* obtained by fixing the last position in the permutations.

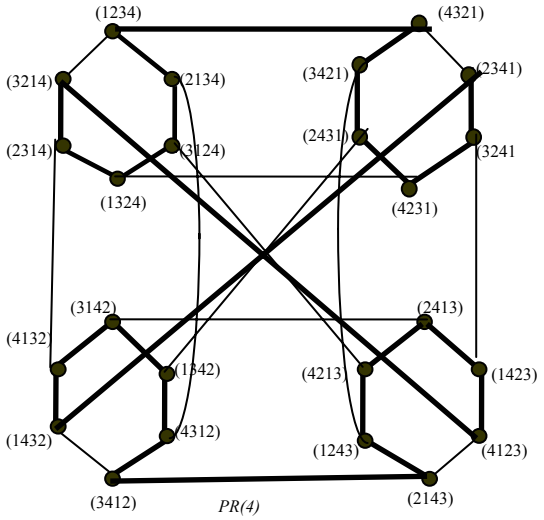

Fig. 1

2.2 Processor ordering

One of the important properties of the pancake graph is that it contains a hamiltonian cycle [4] and this cycle can be constructed as it follows.

Definition. For $k = \overline{2,n}$ the pancake sequence G_k of order *k* is the sequence of generators recursively defined by

$G_2 = g_2$;

For $k>2$, $G_k = \langle G_{k-1}, g_k, G_{k-1}, g_k, ..., G_{k-1} \rangle$ where G_{k-1} occurs *k* times in the sequence.

Proposition. Given a permutation $\pi \in S_n$, for $k = \overline{3,n}$, (π, G_k) defines a hamiltonian cycle over the *k*- pancake containing π.

Example. For the pancake P_4 the vertices ordered by the pancake sequence is

1234→ 2134→ 3124→ 1324→ 2314→ 3214→ 4123→ 1423→ 2413→ 4213→ 1243→ 2143→ 3412→ 4312→ 1342→ 3142→ 4132→ 1432→ 2341→ 3241→4231→ 2431→ 3421→ 4321→ 1234.

and the hamiltonian cycle is shown in figure 1 using bold lines.

3 The Parallel Interpolation Algorithm

The parallel algorithm is based on the algorithm described in [6] for computing a *N= n!* node Lagrange interpolation on a *n*-star graph. We shall apply this algorithm for a network using a pancake topology with bidirectional links between nodes. Let *N=n!* be the number of the nodes in *PC(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 pancake *PC(n)* represented by the permutation $w \in S_n$. 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, $i = \overline{1,6}, w \in S_n$ and by $P_w^{(t)}(R_i), i = \overline{1,6}, w \in S_n$ 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 *PC(n)* durring the main phase.

When constructing a hamiltonian cycle in *PC(n)* as shown in subsection 2.2, two arrays, *Next[w]* and *Previous[w]*, which indicate the nodes before respectively after node *w*, $w \in S_n$ 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 be set to their proper values before starting the initialisation phase.

3.1 Initialisation phase

In this 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. The assignation of the points (x_i, y_i) to the processor is made in such a way that each point is assigned to a processor. For example, the point (x_i, y_i) can be assigned to the processor of order *i* in the ordering presented in subsection 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 $w \in S_n$ in parallel:

$$P_w^{(0)}(R_1) = 1; P_w^{(0)}(R_2) = 1; P_w^{(0)}(R_3) = x_i;$$

$$P_w^{(0)}(R_1) = x_i; P_w^{(0)}(R_5) = x - x_i; P_w^{(0)}(R_6) = x - x_i;$$

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

$$P_w^{(t+1)}(R_3) \Leftarrow P_{Next[w]}^{(t)}(R_3);$$

$$P_w^{(t+1)}(R_4) \Leftarrow P_{\mathrm{Previous}[w]}^{(t)}(R_4);$$

$$P_w^{(t+1)}(R_5) \Leftarrow P_{Next[w]}^{(t)}(R_5);$$

$$P_w^{(t+1)}(R_6) \Leftarrow P_{\mathrm{Previous}[w]}^{(t)}(R_6);$$

$$P_w^{(t+1)}(R_1) = P_w^{(t+1)}(R_1) \times P_w^{(t+1)}(R_5) \times P_w^{(t+1)}(R_6);$$

$$P_w^{(t+1)}(R_2) = P_w^{(t+1)}(R_2) \times (x_i - P_w^{(t+1)}(R_3)) \times \times (x_i - P_w^{(t+1)}(R_4));$$

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 realised in parallel because of bidirectional links between nodes), 2 substractions and 4 multiplications.

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

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

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 star network topology presented in [6].

This phase consists in *(n-1)* subphases, each with *O(log n)* steps, each step including three communication operations and one addition. In total, this phase includes $\sum_{i=2}^{n} \lceil \log i \rceil$ additions and $1 + 3 \cdot \sum_{i=3}^{n} \lceil \log i \rceil$ communication operations.

4 Conclusion

Due to its properties, the pancake graph is an attractive topology for interconnection networks. The pancake graph is a Cayley graph, so it has a lot of desirable properties as vertices and edge symmetry, high connectivity and it is a Hamiltonian graph. It has also lower degree and diameter than the classic hypercube topology.

In this paper, the parallel algorithm that computes a *N=n!* point Lagrange interpolation on a *n*-star graph is applied for a *n*- pancake network. This algorithm, that works in three phases, requires in total $N/2 + 1 + 3 \cdot \sum_{i=3}^{n} \lceil \log i \rceil$ data communication steps, *2N-1* multiplication operations, $2N - 2 + \sum_{i=2}^{n} \lceil \log i \rceil$ substractions or additions and one division. 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 $O(N^2)$.

The main phase of the algorithm works for all the topologies that contain a Hamiltonian cycle. The method used for the final phase depends on the properties of each topology. The property of being Hamiltonian can be used in this phase, but the number of communication steps increases. The property of self-similarity (or recursive decomposition) is much useful in this phase.

This algorithm can be used to compute similar functions, like Hermite interpolation, trigonometric interpolation and others.

References:

[1] B. Akers, B. Krishnamurthy, A Group Theoretic Model for Symmetric Interconnection Networks, *IEEE Transactions on Computers*, vol. 38, no. 4, 1989, pp. 555-566

[2] B. Goertzel, Lagrange interpolation on a tree of processors with ring connections, *JPDC*, 22, 1994, pp. 321-333

[3] C.N. Hung, H.C. Hsu, K.Y. Liang, L.S. Hsu, Ring embedding in faulty pancake graphs, *Inform. Process. Lett.*, vol. 86, 2003, pp. 271-275

[4] C. Lavault, Embeddings into the Pancake Interconnection Network, *Proc. of High Performance Computing on Information Superhighway*, HPC Asia 1997, pp.73-78

[5] C.K. Ling, H.M. Huang, L.H. Hsu, The super connectivity of the pancake graphs and the super laceability of the star graphs, *Theoretical Computer Science*, vol. 339, Issue 2, 2005, pp. 257-271

[6] H. Sarbazi-Azad, M. Ould-Khaoma, L.M. Mackenzie, A Parallel Lagrange Interpolation on the Star Graph, *Proc. 14th IPDPS*, Cancun, Mexico, 2000, pp. 777-782

[7] H. Sarbazi-Azad, M. Ould-Khaoma, L.M. Mackenzie, A Parallel Lagrange Interpolation on k-ary n-cubes, *LNCS1557*, 1999, pp. 85-95

[8] H. Sarbazi-Azad, M. Ould-Khaoma, L.M. Mackenzie, 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, 2000, pp. 593-598

[9] J.J. Sheu, J. Tan, K.T. Chu, Cycle embedding in pancake interconnection networks, *Proc. 23rd Workshop on Combinatorial Mathematics and Computation Theory*, Taiwan, 2006, pp. 85-92

[10] I. Zelina, Parallel Lagrange Interpolation in Extended Fibonacci Cubes, *Studia Univ Babeş-Bolyai*, vol. L, no. 1, 2005, pp. 105-110