Survey on Parallel Algorithms

Shweta S. Bagul
Department of Computer Technology,
Veermata Jijabai Technological Institute,
Mumbai, India

Omkar J. Kulkarni
Department of Computer Technology,
Veermata Jijabai Technological Institute,
Mumbai, India

Abstract-
Researchers have sought cost-effective improvements by building "parallel" computers-computers that perform multiple operations in a single step. The parallel algorithms guide these parallel computers to carry out this task. The scalable performance and lower cost of parallel platforms is reflected in the wide variety of applications. In this talk, we will present various applications of parallel algorithms, challenges associated in designing them.

Keywords-Parallel algorithms, parallel processing

I. INTRODUCTION

Basically, parallel algorithms are those, which perform multiple operations in a single step. There are several different forms of parallel computing: bit level, instruction level, data, and task parallelism. Parallel computer programs are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronization between the different subtasks are typically one of the greatest obstacles to getting good parallel program performance.[1]

Parallel algorithm was necessary & implementable because of substantial improvements in multiprocessing systems & rise of multi-core processors. Performance of a computer is decided by Time required to perform basic operation (limited by clock cycles) & No. of these basic operations that can be performed concurrently. eg: splitting up the job of checking all of the numbers from one to a hundred thousand to see which are primes could be done by assigning a subset of the numbers to each available processor, and then putting the list of positive results back together. Factors Affecting Estimation Of The Complexity/Cost Of Parallel Algorithms:

- Time
- memory(space)
- communication between different processors

This communication is achieved by message passing & shared memory.

The Paper Is Organized As Follows. The Very First Part Contains The Fundamentals Of Parallel Algorithms. After That the Applications of Parallel Algorithms in Various Fields are stated.
The observations regarding Time Complexities Are Illustrated. The second last section has a algorithm proposed by myself which finds out an array of centers of the intervals of real line. The Last Part Contains References.

II. LITERATURE SURVEY

2.1. Applications of Parallel Algorithms

1.1.1 In Computer Organization

In This paper the author describes use of Parallel Dynamic Programming Algorithm on a Multi-core Architecture. Dynamic programming algorithms consider the result from the available results set what is suitable at the given moment of time. There is a data dependency among various stages. It maybe consecutive or non-consecutive. Here We propose a parallel pipelined algorithm for filling dynamic programming matrix by decomposing the computation operators. Dynamic programming is useful in many search and optimization problems.[2]

The author proposes given algorithm in following way. A general strategy on a cache memory model is to develop parallel out-of-core algorithm. In order to facilitate the study of the methodology for designing algorithms on such a large scale multi-core architecture, it is necessary to build a programming/execution model.

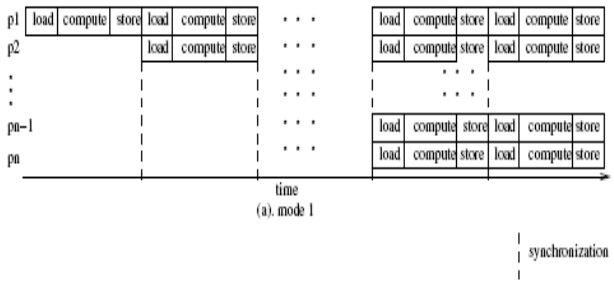

Fig.1 The execution model of previous out of-core model

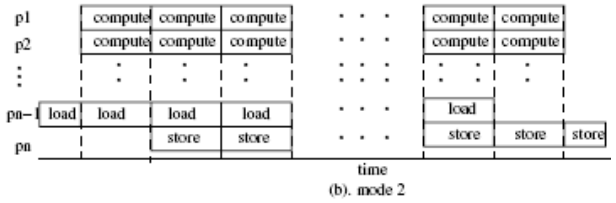

Fig.2. The execution model of out-of-core model on multi-core architecture. The number of helper thread depends on the architecture parameters such as bandwidth

The Parallel pipelined algorithm is discussed further. Let us assume that we have p + 2 threads, two of which are helper threads, and the size of transformed domain (DP matrix) is n. Because of the data dependence between two consecutive entries in the same row and column, we cannot get efficient parallelism. Hence use decomposition. During computation of the second part, the sub-matrices A(i, i) and A(j, j) are triangular. The two operations A(i, i)⊗ A(i, j), A(i, j)⊗A(j, j) depend on the final results of A(i, j), so A(i, i), A(j, j), A(i, j) are integrated into one sub-matrices, where the parallelism can be exploited along the diagonal. The computation of one block which is divided into 4 sub-blocks as below-

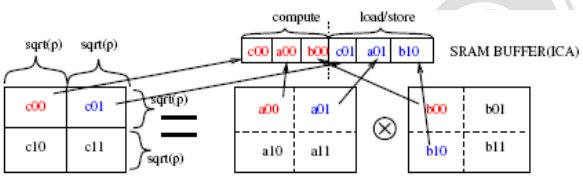

Fig.3 Execution of a single block

This model is an extension conventional out-of-core model, therefore our proposed algorithm can be adapted to achieve high performance on conventional out-of-core model.

2.1.2. In Mobile Communications

The parallel algorithms can be used in Real-Time Generation of Bit-Wise Parallel Representations of Over-Sampled PRN Codes. A significant computational challenge for a real-time code-division multiple access (CDMA) software receiver are to perform baseband mixing, Pseudo-random number (PRN) code mixing, Accumulation of the resulting correlations. Fast execution can be achieved by storing signals in a bit-wise parallel format and using bit-wise logical operations such as AND and EXCLUSIVE OR to process multiple data samples simultaneously.[3] Such methods increase processing speed by a factor of 2 to 4 if the raw radio-frequency (RF) data and the mixing signals are 1- or 2-bit digital signals.

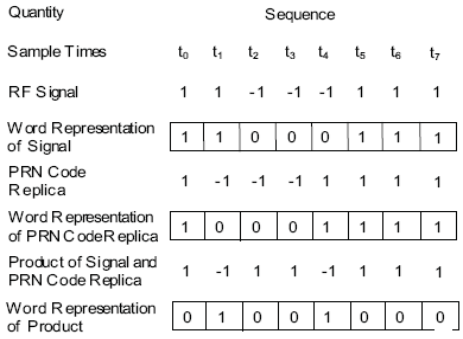

Fig. 4 An 8-bit example of bit-wise parallel storage and mixing of a radiofrequency signal and a PRN code replica

The Sampling Example is given as below. Eight samples of a 1-bit digital RF signal get stored in parallel in a single 8-bit integer word. Corresponding PRN code replica is stored in parallel in another 8-bit integer word. An EXCLUSIVE OR operation accomplishes parallel 8-sample mixing of the two signals to produce the 8-bit word at the bottom of the figure. This letter is concerned with efficient generation of the bitwise parallel over-sampled representation of the PRN code in the middle word of Fig. Although not provably optimal, its technique is practical and enables real-time generation of such representations. Over-sampling and translation into a bitwise parallel representation are accomplished using tabulated functions. Over sampling and bit wise parallel storage of prn codes is possible.

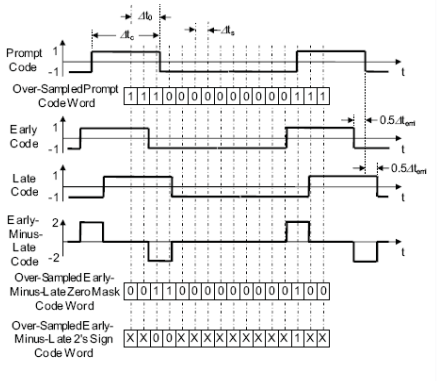

Fig.5 The CDMA code sample

CDMA PRN codes are sequences of +1 and -1 chip values that chip at the rate

$$fc = 1/\Delta tc$$

Many receivers work with prompt and early-minus-late (EML) replicas of a PRN code. The early chips start $0.5\Delta t_{eml}$ seconds before the prompt chips, and the late chips start $0.5\Delta t_{eml}$ seconds after the prompt chips. It needs prompt and EML code replicas sampled at the RF data sample times, which are depicted in Fig. as vertical dash-dotted lines. The receiver's sampled PRN codes

are called over-sampled because $\Delta t_s < \Delta t_c$, which implies that $f_s > f_c$.

The sampled values of the prompt and EML PRN codes can be represented in bit-wise parallel formats as integer words.The prompt code is represented by a sign bit with a 1 bit representing +1 and a 0 bit representing -1. The representation of the prompt code at the 16 sample times of Fig. Starts with three 1s, continues with ten 0s, and finishes with another three 1s. Its 16-bit unsigned integer word representation is

215 + 214 + 213 + 22 + 21 + 20 = 57351.

Table construction requires knowledge of the maximum possible number of code chips needed in order for the early, prompt, and late codes to span an entire data word of ns samples. This maximum is:

$$L = \frac{\text{floor}[n_s - 1]\Delta t_s - \Delta t_{0kmin} + 1/2\Delta t_{eml}}{\Delta t_c}$$

The size of each table can be determined from the parameters kmin, kmax, and L. The Efficient Table Look-Ups Of Over-Sampled Codes During An Accumulation is given by following steps. A typical set of accumulation calculations in a software receiver starts when prompt chip C(1) starts and ends when prompt chip C(M) ends.Timing relationship between data sample words and the sequence of prompt code chips that define an accumulation interval.

The accumulation interval starts Δt_{start} seconds past the first sample of data word W1 and ends $M\Delta t_c$ seconds later during data word WN, which implies that

$N = \text{ceil}[(\Delta t_{start} + M\Delta t_c)/(ns\Delta t_s)]$.

Additional zero-masking is required if the first samples of data word W1 or the last samples of data word WN do not lie within the accumulation interval.

The Performance of Algorithm is given as follows-

1. Memory Requirements :

The total number of bytes required –

Memory = $3\text{ceil}[ns/8].2^L k_{tot}$ bytes

2. Operations Count:

The critical operations count for this method is the number Of integer operations per processed data word. Brute-force PRN code over-sampling requires 6 operations per sample when used in integer accumulation calculations and 9 operations per sample if the accumulations are calculated using bitwise parallel computations.

Approximation-Induced Code Distortion:

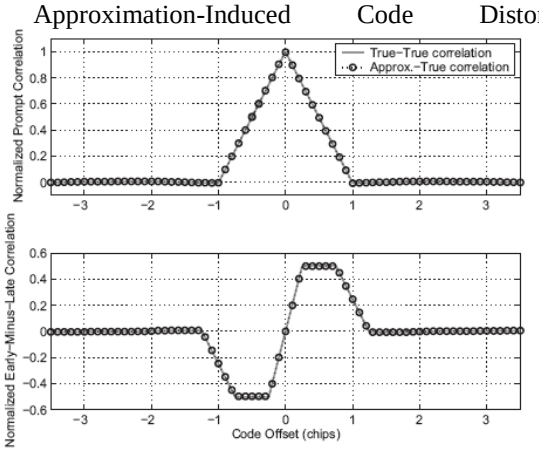

Fig.6 Distortion graph of Early-minus-late correlation vs.normalised prompt correlation

The low distortion of the new method is illustrated by the example correlation plots in fig. A new method has been developed for use in a real-time software radio receiver that uses bit-wise parallel operations to perform the de-spreading correlations required to acquire and track CDMA signals. The method tabulates all possible PRN code chip sequences on a grid of possible code timing offsets in order to produce all possible versions of the sampled code within a single data word. A recursion computes the proper indices based on timing relationships between data samples and PRN code chips.

2.1.3. In Routing

In this paper author suggests Parallel Routing Algorithms for nonblocking Electronic and Photonic Switching Networks.

To build a large IP router with capacity of 1 Tb/s and beyond, either electronic or optical switching can be used. The deployment of optical fibers as a transmission medium has prompted searching for the solution to the problem of speed mismatching between transmissions and switching. A hybrid

approach in which optical signals are switched, but both switch control and routing decisions are carried out electronically is required.[4] A switching network usually comprises a number of switching elements, grouped into several stages interconnected by a set of links. Any routing algorithm requiring more than linear time would be considered too slow. A class of multistage nonblocking switching networks has been proposed. In this class, each network, denoted by B(N, x, p,α) has relatively low hardware cost and short connection diameter in terms of the number of SEs. A B(N, x, p,α) is constructed by horizontally concatenating x(≤ lg N-1) extra stages to an NxN Banyan-type network, and then vertically stacking p copies of the extended Banyan.B (N, x, p,0) and B(N, x,

p,1) are similar in structure, but the latter does not allow any two connections with the same wavelength passing through the same SE at the same time while the former does.

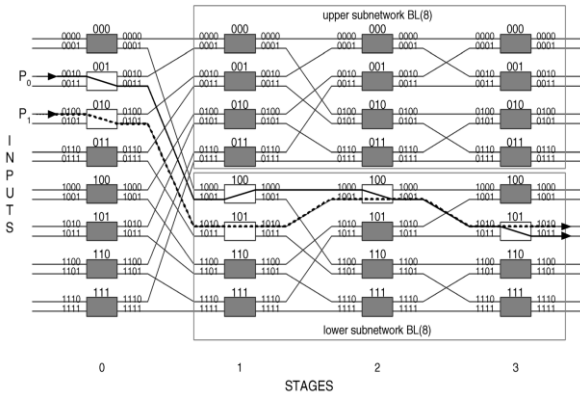

Fig. 7 Self-routing connection paths P0 and P1 in BLð16Þ with link and node conflicts

Designing Parallel Switch Routing Algorithms

A trivial lower bound on the time for routing K K(0 ≤K≤N) connections sequentially in B(N, x, p,1) is Ω(K lgN). This lower bound is obtained by assuming that for any connection it takes O(1) time to correctly guess which plane to use without conflict and O(lgN) time to compute the connection path in that plane.

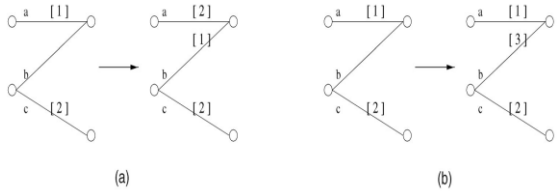

fig. 8 (a) A (weak) edge-coloring. (b) A strong edge-coloring

The major contribution of this paper is the design and analysis of parallel routing algorithms for a class of non blocking switching networks, B (N, x, p,α). Although the assumed parallel machine model is a completely connected multiprocessor system of N PEs, the proposed algorithms can be transformed to algorithms for more realistic parallel computing models.

Time complexity = O (lgN lg lgN)

2.1.4. In Graph Theory

Here, author proposes a Parallel Algorithm for Enumerating All Maximal Cliques in Complex Network. Many Networks In Our World Are Complex Networks Involving Massive Data. Efficient enumeration of all maximal cliques in a given graph has many applications in Graph Theory, Data Mining and Bioinformatics. This enumeration can be done parallel which takes optimum time than the sequential algorithms. To solve the maximal clique problem in the real-world scenarios, this paper presents a parallel algorithm Peamc (Parallel Enumeration of All Maximal Cliques) which exploits effective techniques to enumerate all maximal cliques in a complex network.[5] First we will find what is a clique.

For the graph G, V(G) and E(G) denote the set of vertices and edges of G. A complete sub-graph of G is called a clique. If a clique is not contained in any other cliques, this clique is called a maximal clique. Since that the triangle structure or 3-clique is a basic sub-structure of any clique whose size is larger than 3.It has a close relationship with the clustering coefficient property of the complex network.

We could take advantage of this fact to design our traversal policy for the vertices which are contained in triangles by depth-first order, so that the maximal cliques can be detected with efficiency. Because each search tree rooted with every vertex in G will be traversed, all the candidate cliques will thus be identified. Peamc employs a pruning policy explained as follows.

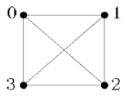

Fig.9. **A 4-clique**

In Figure, we start from node 0 and choose node 1 from *τ(0). After {0,1,2,3} is detected, we continue to* choose node 2. However, we find {0,2,3} is not a maximal clique. This traversal contributes nothing. Consequently, we add a new set η(v0) that caches the maximal cliques obtained from the search tree rooted with v0. After σ(v0) is calculated, we will first merge the current candidate clique with σ(v0) and check whether η(v0) contains the merged set. If so, the merged set will not be a maximal clique and the next traversal steps starting from the current candidate clique will be pruned.

Algorithm Peamc can be explored as below.

For the graph G, V(G) and E(G) denote the set of vertices and edges of G. For a vertex v, set τ(v) represents the neighbors of v.

Since v has |τ(v)| neighbors and these neighbors could constitute α=|τ(v)|×(|τ(v)|-1)/2 triangles at most.

Here We have assumed that,

β= the actual number of triangles

ε =(β/α) The Clustering Coefficient of v

σ(v) = To store all the vertices that could constitute triangles with v and its neighbors.

The algorithm is given as

Input: a large sparse graph *G = (V, E);* Output: the complete set of all maximal cliques; Method:

1: Read graph *G*

2: Generate set *τ(v) for every vertex of G*

3: for each vertex *v of G* 4: call recursive_find_cliques({*v*}, *τ(v))*

5: end for

Function: recursive_find_cliques (*x, τ)*
7: for each vertex t∈*τ by the ascending order*
8: calculate set *σ*
9: if σ≠ Ø
10: extend *x with t*
11: call recursive_find_cliques (*x, σ)*
12: end if
13: else // **Theorem 1**
14: end for

III. OBSERVATIONS

The time complexity of Parallel Dynamic Programming Algorithm on a Multi-core Architecture is found to be $O(n^3/\sqrt{p})$,In graph theory it is O(log N) whereas in the application of nonblocking Electronic and Photonic Switching Networks it is O (lgN lg lgN)

IV. PROPOSED ALGORITHM

I have taken the help of all the above studied algorithms and proposed one algorithm which finds out the centres of the intervals on the real line. We consider the two end points of the interval as first(I)& last(I).I have then divided these into no. of small intervals & found the centre of each interval using parallel algorithm.[7]

Algorithm: Parallel-Find-Center (I);
Input: a family $I = \{Ii = [ai,bi] \mid ai \leq bi, 1 \leq i \leq n\}$ of intervals on the real line;
Output: the center of the corresponding interval graph if the center exists
1. begin
2. let K be the distance between first(I) & last(I) and di_1, be the distances from first(I) to Ii; for $1 \leq i \leq n$
3.if K = -1 then return(0);
4.construct the family of intervals I' = { Ii = [-ai,bi] | ai ≤ bi, 1 ≤ i ≤n}
let d_{i2}, be the distances from first(I') to Ii' : for 1 ≤ i ≤n
5. for each *Ii* (1 ≤ i ≤n) do in parallel
6. if max{d_{i1},d_{i2}}=[K/2] then
7.mark I*i*;
8.Record all the marked intervals in the array C;
9.Return(C)
10.end; //Parallel-Find-Center

V. CONCLUSION

In this way we have studied the versatile use of parallel algorithms in many emerging fields. The parallel algorithms can solve the problems in various domains such as mobile communications, routing & switching as well as route optimization and graph theory.

VI. REFERENCES

[1] Introduction to parallel & distributed algorithms by Carl Burch www.toves.org/books/distalg/
[2] Guangming Tan, Ninghui Sun, Guang R. Gao," A Parallel Dynamic Programming Algorithm on a Multi-core Architecture", ACM journal
[3] Mark L. Psiaki," Real-Time Generation of Bit-Wise Parallel Representations of Over-Sampled PRN Codes", IEEE journal
[4] Enyue Lu, S.Q. Zheng," Parallel Routing Algorithms for nonblocking Electronic and Photonic Switching Networks", IEEE TRANSACTIONS
[5] Nan Du, Bin Wu, Liutong Xu, Bai Wang, Xin Pei," A Parallel Algorithm for Enumerating All Maximal Cliques in Complex Network" , IEEE International Conference on data mining
[6] www.google.com
[7] www.wikipedia.com