Parallel Random Walk Algorithm in VLSI Analysis

Guo Jun
Dept. of Computer Science and Technology
Northwest University
Xi'an, China
e-mail: guojun@nwu.edu.cn

Zhang Cangsong, Cui Jiao
Dept. of Computer Science and Technology
Northwest University
Xi'an, China
e-mail: cuijiao1988@163.com

***Abstract*—Parallel computing techniques were introduced to improve random walk algorithm. A formal model was firstly adopted to explain the random walk problem. And then, the parallel features of random walk algorithm were discussed in detail. Finally, a parallel random walk algorithm was proposed and applied to analyze the VLSI power grid. Experiments were completed at a parallel computing environment of IBM blade computer. Time complexity and the main factors impacting on the execution time of algorithm were analyzed carefully. The experimental results proved that the parallel computing techniques could improve of random walk algorithm effectively. The speedup ratio is close to the maximum value.**

Keywords-parallel computing; random walk; algorithm design

I. INTRODUCTION

Random walk algorithm is an approximation algorithm, belonging to Monte Carlo method. For some NP-hard problems or problems with infinite solution space, random walk algorithm is much more efficient. Therefore, it has been applied to many engineering areas, such as bioinformatics, computational physics, VLSI (very large scale integrated circuit) analysis, macro-economics and so on.

Nowadays, parallel computing technology is becoming more popular than ever, which provides opportunities to improve the existing algorithms including random walk algorithm. This paper thus focused on parallel random walk algorithm and discusses its application in VLSI analysis.

In the past years, some efforts were spent to improve the random walk algorithm in electrical circuit design [1-6]. Reference [2] proposed a partial random walk algorithm for P/G network analysis combined with the SOR technology. Reference [5] proposed a hierarchical random walk algorithm and to the P/G network with regular topology and more power nodes. Reference [6] proposed a pseudo-parallel random walk algorithm for non-parallel computing environment. In this paper, we propose a real parallel random walk algorithm supported by cluster computing system with message pass interface (MPI).

In this paper, we will firstly introduce the basic concepts of random walk algorithm and its application in circuit analysis, and describe the basic theory of parallel computing as well. And then discuss the parallelization features of random walk algorithm. Finally, we will scheme out the parallel random walk algorithm and implement it. VLSI analysis experiment by parallel random walk will carry out in IBM blade computer. Experimental results will be analyzed to derive our conclusions.

II. RANDOM WALK ALGORITHM

A. Random walk problem

Random walk is a classic statistics problem, which can be described formally in figure 1 [7]. A walker is lost in a city with a number of roads and intersections. He starts from one of the intersections, and selects one road randomly to the next intersection. Different roads may be selected according to different probability. While passing an intersection, the walker needs to pay some money as his traveling fare. When the walker gets home, his destination, he can get some reward and ends this walking trip. In this game, there is more than one destination, and the reward of destination is also different. Supposing the walker starts from a certain intersection, he will get some reward after several random walking trips. The final question we must shoot is how to obtain the expectation value of reward, that is, the statistical average of reward.

According to the principles of probability and statistics, starting from node x, the expression of reward from one walking trips is as follows [5]:

$$E(x) = \sum_{i=1}^{\deg ree(x)} p_{x,i} E(i) - m_x \tag{1}$$

Where, $p_{x,i}$ is the probability of walking to the ith adjacent intersection from intersection x, obviously $\sum_{i=1}^{\deg ree(x)} p_{x,i} = 1$; $E(i)$ is the expected reward of random walking trips starting from the *i*th neighboring intersection; $\deg ree(x)$ is the degree of x and equals to the number of adjacent sides with x; m_x is the fare at a certain intersection.

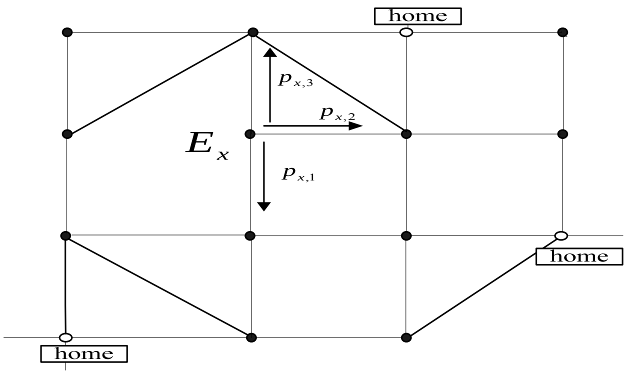

Figure 1. Random walk problem

The key to random walk problem relays on two basic rules: the law of large numbers (LLN) and Central Limit Theorem (CLT). We set x_1, x_2, x_3 ... as the rewards of a great number of random walking experiments. Then the average of these results is expressed as:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2}$$

The variance of these results is expressed as:

$$\delta^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n} \tag{3}$$

We specify the allowable error is Δ, and the probability of error between [-Δ, +Δ] is P, which is called confidence probability. Then, the minimum number of random walking trials is *M*, which satisfies [7]:

$$M > \left(\frac{\beta \delta}{\Delta} \right)^2 \tag{4}$$

Where β is a coefficient which is related with the confidential probability and it can be obtained through normal distribution sub-bit. *M* is also the condition of ending random walk experiments. Now, the average value we get from experiments can be used as the approximate value of $E(x)$.

B. Random walk in circuit network

Currently, the integration of VLSI scaled up a lot, the corresponding circuit equations are becoming much more larger. So traditional matrix solution, relaxation iteration method or circuit simulators (such as HSPICE) are unable to meet the needs for analyzing the circuits of the chip. The random walk algorithm is one of the candidates for VLSI analysis, and it is mainly used in the reliability analysis of P/G (Power/Ground) network.

The P/G network in VLSI is usually modeled as resister network. The node structure of P/G network shows Fig. 2. Appling Kerchief's current law and Ohm's law to the circuit, an equation could be derived as below:

$$g_1(V_1 - V_x) + g_2(V_2 - V_x) + g_3(V_3 - V_x) + g_4(V_4 - V_x) = I_x \tag{5}$$

Then, we could derive the voltage expression of node x as follow:

$$V_x = \sum_{i=1}^{degree(x)} \frac{g_i}{\sum_{i=1}^{degree(x)} g_i} V_i - \frac{I_x}{\sum_{i=1}^{degree(x)} g_i} \tag{6}$$

Define g_i as the conductance of the *i*th connected components; $\deg ree(x)$ is the degree of node x and equals to the number of adjacent components with x; V_i is the voltage of the *i*th adjacent node. Equation (6) is a form of harmonic function, and it is similar with the structure of (1). That is, the circuit network can be viewed as a random walk map. And in it, the network nodes correspond to the intersections and the node voltage corresponds to the average of reward, which can be calculated from a deal of random walking trips starting from this intersection. PAD nodes correspond to home nodes, and the reward obtained from home node equals to the voltage of PAD.

According to the rule of random walk, in the circuit network, the probability from node x to adjacent node through *i*th component is:

$$p_{x,i} = \frac{g_i}{\sum_{i=1}^{degree(x)} g_i} \tag{7}$$

The traveling fare at node x is expressed as:

$$m_x = \frac{I_x}{\sum_{i=1}^{\deg ree(x)} g_i} \tag{8}$$

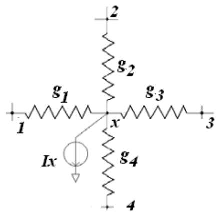

Figure 2. The structure of node x

Therefore, as long as establish a corresponding random walk circuit network roadmap, we can use random walk technique to calculate the voltages of the network nodes without complex matrix operations.

III. PARALLEL RANDOM WALK

A. Parallel Computing

Usually, parallel computing is based on high performance computers, including distributed network computers, parallel computers, distributed computers and other high-performance computing systems. There are two types of parallel computing technology: time parallel and space parallel. Time parallel use assembly line technology while space parallel refers to concurrent computing on multiple processors. Here we just discuss the space parallel technology. This technology is supported by multi-processor computer or multi-computer system. Generally speaking, a large complex problem will be divided into a number of smaller sub-problems, which will be assigned to the appropriate processors of the parallel computer. All sub-problems will be process concurrently in parallel computing systems and the procedure of trouble shooting is speedup.

Parallel program usually works at master-slave mode. A master process and n sub-processes should be created in initial part of the program. The main process plays as the master responsible for broadcasting the process node information and calculating the values from child processes. The child processes play as slaves and carry out the assigned tasks. Using MPI_Send function, child processes send the results back to the main process, and the main process uses MPI_Recv function to receive data and work out the final result.

There are two important factors to effect the performance of a parallel program. The first one is speedup, which indicates how many times the execution speed of parallel algorithm faster than the serial algorithm for a given application. Under the premise of the isomorphic parallel machines and exclusive processor resources, the execution time of a serial application with a single processor is T_s. Meanwhile, if the serial application is divided into P processes and the execution time with P processors is T_p. Then, the speedup S_p is defined as:

$$S_p = T_s / T_p \tag{9}$$

Clearly, the speedup of a parallel program is determined by the slowest process.

The second one is efficiency, which indicates the efficiency of the parallel program, is defined as:

$$E_p = S_p / p \tag{10}$$

B. Parallel random walk algorithm

The random walk algorithm has inherent parallel characteristics because every trip and every step are probability independent events, which could be carried out concurrently [7].

Figure 3 shows a random walk trip for a specified node N_0. Starting from N_0, the walker passed node N_1, N_2, N_3 ...N_i, and get home node, pads in P/G networks. For every passed node N_k, the corresponding numbers of visits were V_k, and the reward is W_k , 0< k<=i. Then, the total reward obtained from this random walk trip for node N_0 is expressed as:

$$W_0 = \sum_{k=1}^{i} W_k * V_k \tag{11}$$

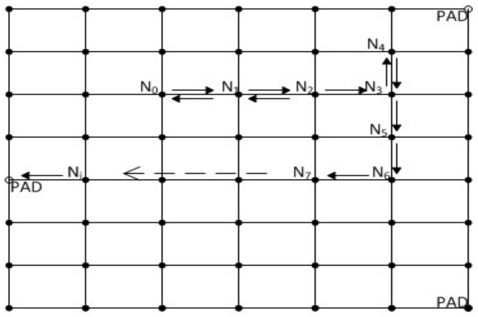

Figure 3. Random walking trip from N_0

At the same way, we can start a new random walk trip for node N_0 and get a new reward W_1. The random walk experiments will continue until the results satisfy expected value.

Considering figure 2 as a circuit structure of node N_x. Equation (7) shows that the probability of visiting each component has remained unchanged and the circuit topology would not be changed as well. For every step in a random walk trip, the program generates a random number to determine which node will be visited next step. And there is no relationship between two walk trips. They are concurrent task.

Figure 4 describes random walk trips starting from two different nodes M_0 and S_0. Although two trips might pass some common nodes, they are not relevant. That is to say, random walking trips for different nodes are independent too, they are concurrent tasks.

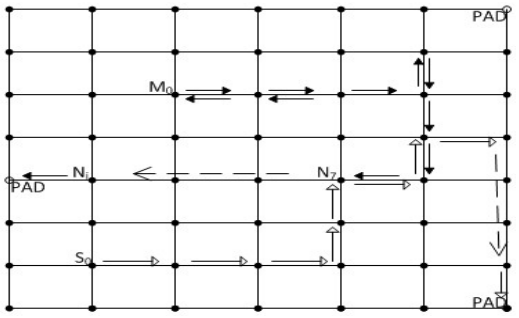

Figure 4. Random walking trips from M0 and S0

Currently, the mainstream parallel computers include SMP (shared memory symmetric multi-processor), MPP (large-scale distributed memory parallel machine), DSM (distributed shared memory multi-processor) and COW (cluster of workstations). COW is the more popular parallel architecture [8], our experiments were done on COW.

We use the average division strategy in the parallel random walk algorithm. Assuming n processors are available, the random walk of K nodes will be divided averagely. So each processor performs K/n nodes' random walk tasks. At the same time, we can also divide one node's M random walking trips averagely, and each processor will perform M/n random walk trips. All computing tasks assign to multiple processors and execute concurrently. After all processors finish a certain number of random walk trips, they communicate through MPI to determine whether the random walking should be ended.

IV. EXPERIMENTAL RESULTS

The parallel random walk algorithm is implemented in C language. The program run on IBM Blade HS22 server, which contains 4 computing nodes, each computing node includes 4 Intel Xeon QC E5630 CPU 2.66GHz. We designed several circuit structures with different PAD proportions and scales, and calculated the voltage of all the nodes and single node, setting the voltage error ranging from -0.5 to 0.5 and recording the calculation time (s). Parallel speedup and efficiency are used as standards to evaluate the performance of this parallel algorithm program. The results of experiments are shown in table I.

It is clear that parallel algorithm is much faster than serial algorithm. However, according to Table Ⅰ, with the increase of process, the parallel efficiency decreases, which is due to the unbalanced load distribution among different processes. As some nodes are closer to PAD nodes, they end the random walking trips faster. But for nodes far from the PAD, walking time is relatively longer. Besides, more processes take more communication cost.

Table I also shows that the runtime of parallel random walk increased with the circuit scale growing up. In our experiments, when the network scale has been increased to 400*400, the parallel efficiency of parallel random walk algorithm is closer to 100%, Seen from Table I. In fact, when the size of integrated circuits is up to a certain magnitude, traditional simulation tools can not solve this problem, or the solution time is impossible to endure. However, the parallelization method of the random walk algorithm is feasible to solve this problem. In a word, parallel random walk algorithm is effective to solve large-scale circuit voltage, and with the expansion of the circuit, the parallel efficiency of parallel algorithms is closer to the ideal value.

V. CONCLUSION

We present a parallel random walk algorithm and applied it to solve P/G grid. The experimental results show that parallel technology dramatically improves the speed of random walk algorithm, and the speedup will be close to the ideal value in the best case. Compared with other improved technologies about random walk algorithm, parallel technology does not need to modify the circuit structure and relatively easy to achieve. This method is adaptable one and is a great potential improved technology for random walk algorithm.

REFERENCES

[1] H. Qian, S.R. Nassif,and S. S. Sapatnekar. Random Walks in a Supply Network. In Proceedings of the ACM/IEEE Design Automation Conference, pp. 93-98, 2003.

[2] Zuying Luo, Guopu Wang, Yici Cai. part. Fast solution methods for P/G network based on partical random walk. In Journal of Computer-Aided Design and Computer Graphics, pp.1535–1541, 2004.

[3] Junyong Deng, Jianghua Qian, Cheng Zhuo, Jinfang Zhou, Kangsheng Chen. Improved random walk algorithm for P/G network analysis. In Journal of Zhejiang University (Engineering Science).pp. 1324–1328, 2007.

[4] Haohang Su, Yimen Zhang, Yuming Zhang, Min Xie, Jincai Man. Simulation based on an improved compression algorithm for static random walk P/G. In Chinese Journal of Computational Physics, pp. 673-676, 2007.

[5] H. Qian, S. S, Sapatnekar. Hierarchical random walk algorithms for power Grid analysis. In 10th Asia South Pacific Design Automation Conference, pp. 499-504, 2004.

[6] Jun Guo, Sheqin Dang, Satoshi Goto. Random Walk Algorithm for Large Thermal RC Network Analysis. In IEEE 8th International Conference on ASIC, pp. 771-774, 2009.

[7] Jun Guo, Minghui Li, Sheqing Dong, Weichang Shen. Random walk for circuit analysis and improving by parallel computing. In Computer Engineering and Applications, pp.199-201, 2010.

[8] Xiong Hong, Guangming Dai, Chunxia Leng. Construct COW Based on MPICH in Linux Environment. In Control and Automation, pp. 124-126, 2006.

TABLE I. EXPRIMENTAL RESULTS

Process number	Circuit Scale	Parallel runtime(s)	Serial runtime(s)	Speedup	Efficiency
2	50*50	198.187311	375.20553	1.89	94.7%
	100*100	14951.03688	28556.48045	1.91	95.5%
	200*200	1137071.5665	2179402.3344	1.92	95.8%
	400*400	85045876.7326	163458175.08	1.92	96.1%
4	50*50	99.121948	375.20553	3.79	94.6%
	100*100	7499.075748	28556.48045	3.81	95.2%
	200*200	569937.3887	2179402.3344	3.82	95.6%
	400*400	42390494.57053	163458175.08	3.86	96.4%
8	50*50	49.906883	375.20553	7.52	94.0%
	100*100	3764.71281	28556.48045	7.59	94.8%
	200*200	286469.9262	2179402.3344	7.61	95.1%
	400*400	21440049.51207	163458175.08	7.62	95.3%
16	50*50	25.168053	375.20553	14.90	93.2%
	100*100	1890.65681	28556.48045	15.10	94.4%
	200*200	143542.8197	2179402.3344	15.18	94.9%
	400*400	10720074.7560	163458175.08	15.25	95.3%