International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:2, No:3, 2008

Using Multi-Thread Technology Realize Most Short-Path Parallel Algorithm

Chang-le Lu, and Yong Chen

***Abstract*—**The shortest path question is in a graph theory model question, and it is applied in many fields. The most short-path question may divide into two kinds: Single sources most short-path, all apexes to most short-path. This article mainly introduces the problem of all apexes to most short-path, and gives a new parallel algorithm of all apexes to most short-path according to the Dijkstra algorithm. At last this paper realizes the parallel algorithms in the technology of C # multithreading.

***Keywords*—**Dijkstra algorithm, parallel algorithms, multi-thread technology, most short-path, ratio.

I. Introduction

WITH the development of computer, the graph theory research obtains widely takes, and the most short-path question as a model question of graph theory is already applied in many fields. Because the efficiency of the existing most short-path serial algorithm is not very high, and it is already with difficulty in satisfying the need in modern. Therefore, this paper proposes a new most short-path parallel algorithm, and has carried on it using the multi-thread technology.

II. The Question of All Apexes to Most Short-Path

All apexes to the most short-path are calculating all apexes to the most short-path in the graph. In the research of the most short-path question, the Dijkstra algorithm is one of most classical algorithms [3]. In this introduced first the serial Dijkstra algorithm, then further introduces its corresponding parallel algorithm.

A. Serial Dijkstra Algorithm [2]

Supposes an oriented graph $G(V,E)$, W is the weighting adjacency matrix of $|V|=n$, and using $w(i,j)$ expresses the Length of side of (i,j). If between i and j it does not exist the oriented side, then $w(i,j)$ is ∞,

Manuscript received September 8, 2006.

Chang-le Lu is with School of Computer Technology and Automation of Tianjin Polytechnic University, Tianjin, China (corresponding author, e-mail: luchangle_1815@sina.com).

Yong Chen is with School of Computer Technology and Automation of Tianjin Polytechnic University, Tianjin, China.

$1 \le i, j \le n, i, j \in V, S(n)$ was the most short-path end point set array which V_i embarked from the apex $(1 \le i \le n)$. In the array each set original state is a null set. The starting value of from V_i embarking to other apexes in the chart $V_j (1 \le j \le n, j \ne i)$ is $d[j] = arcs[locateVex(G,v)[j]]$, $V_j \in V$.

B. Based on Dijkstra Algorithm Transformation to a New Parallel Algorithm

Because in above Dijkstra algorithm, in the chart each apex sequence of operation is same, therefore it may use the method of the serial algorithm direct parallelization to transformation to the parallel algorithm [1]. Namely: Assigns a processor each apex computation duty to complete separately, it realizes the Dijkstra algorithm parallelization. It is following the new parallelization algorithm:

Input: Oriented graph $G(V,E)$ weighting adjacency matrix $W = \{W_{ij}\}, i, j \in V$

Output: All apexes to most short-path matrix $D = \{d_{ij}\}, i, j \in V$

```
Begin
D←W
For each k par-do
S (k) =k
        d←D(k)
        For i=1 to n do
            m= (min (d (p)) and p ∉ S (k))        1≤p≤n
            S (k) =S (k) +m
            For j=1 to n do
            If j ∉ S (k) and m+W(p,j)< d(j)then
                D [j] =m+W(p,j)
                Endfor
            Endfor
            D (k) ←d
    Endfor
End
```

C. Algorithm Performance Analysis

In the Dijkstra serial algorithm, it has three nestlings cyclic sentences, therefore its time order of complexity is $O(n^3)$; In

the Dijkstra parallel algorithm, the design strategy of serial algorithm direct parallelization has already carried on the first cyclic sentence the resolution, and hands over separately them in the different processor to complete. Therefore, its time order of complexity was $O(n^2)$, and the performance of Dijkstra parallel algorithm will be much higher than its serial algorithm.

III. The Example of between all Apexes most Short-Path in Chart

Using above solution of all apexes to the most short-path solves in the oriented graph which as shown in Fig. 1 all apexes to between the most short-path. First according to the oriented graph apex and the side information which gives, we write the oriented graph the adjacency matrix. The Dijkstra algorithm introduced above is used to solving this question. As space is limited, only aims at the new parallel algorithm here are making a simple analysis using the multi-thread knowledge.

A. Using Multi-Thread Technology Realization Parallel Algorithm Feasibility Analysis

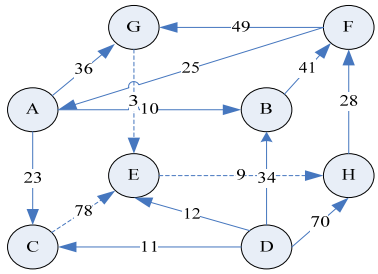

Fig. 1 Example oriented graph

In multi-processor system, because has many CPU hardware resources, so in each advancement rank the independent thread may hold the different CPU resources mutually to achieve in the true sense concurrent execution separately. Possibly it is more than in the multiprocessing system in the advancement rank many concurrent execution threads number the CPU integer, therefore it will create the partial threads inevitably the movement because of the hardware resources limit, and then it will be unable to do the immediate execution. Actually which thread carry out first supports the multi-duties the operating system to carry on the management, does not need to carry on the artificial intervention again, so this greatly reduced the programming work? Because the thread opening and the management are finished by operating system, therefore it will consume extra part of system resources [4]. Might as well the located time section in i, opens M_i thread, these parallel thread computation load respectively records is: Q_{ij} $(j=1,2,\cdots,M_i)$ in remains unresolved question there is n such time section. The operating system manages each thread the time expenses is: t. The total time needs which using the multi-thread computation records is T_j.The ordinary serial procedure computation needs the time records is T. The number of CPU the multi-processor system has is C. Then leaves mutually the independent thread procedure and between not the ordinary serial procedure movement time has the following relations, like the formula (1), formula (2) shows:

$$T_j = \sum_{i=1}^{n} \left(\frac{\sum_{j=1}^{m_i} Q_{ij}}{C} + M_i \times t \right) \quad (1)$$

$$T = \sum_{i=1}^{n} \sum_{j=1}^{M_i} Q_{ij} \quad (2)$$

Not difficult to discover by the above equation that, when in the computer system the processor integer is more than one, and each parallel thread assignment computation load quite is big, and then $\frac{\sum_{j=1}^{M_i} Q_{ij}}{C} + M_i \times t < \sum_{j=1}^{M_i} Q_{ij}$. At This time the efficiency of using the multi-thread design parallel procedure compare to carry out the efficiency the corresponding serial procedure is higher. When the value of $\frac{\sum_{j=1}^{M_i} Q_{ij}}{C}$ is bigger than $M_i \times t$, $M_i \times t$ may neglect regarding the formula (1) influence. At this time $\frac{T}{T_j}$ ratio tends to C. In the actual, the parallel question scale which will be resolved generally is all big, thus theoretically realizing the parallel algorithm using the multi-thread technology is feasible, and can improve the efficiency.

B. Realizes the Dijkstra Parallel Algorithm using the Multi-Thread Technology

In the Dijkstra parallel algorithm, it needs n processors (n is in chart apex number) can satisfy its algorithm request. Also it is changing in the actual process number along with the question. Therefore it is very difficult each time to be able to satisfy the algorithm completely the request (hardware request). In order to achieve versatility of the algorithm, according to the above theoretical analysis again, it simulates in the reality using the multithreading the processor. Therefore when carries on this algorithm the design, it does not need again to consider the concrete hardware equipment, and holds each thread to the system processor the situation to give the operating system to complete, and this step to the user is

transparent. Because the example altogether has 8 apexes, according to above analysis, it should correspondingly assign 8 threads to it to simulate the corresponding processor. Each thread partial variable simulates each processor independent memory. Finally carry on each thread and the computed result compiles, like this then obtain the computed result which finally needs.

IV. Experimental Environment and Test Result

In order to confirm the theoretical analysis which the preamble states, we has performed two experiments for this. Two experimental backgrounds are the same (by this guarantee research question nature invariable), only difference is the environment of the experiments. Its concrete experimental background and experimental environment as follows shows:

A. Experimental Background

In order to achieve the ideal realization effect, in the condition of not changing serial and parallel algorithm design original intention premise, gave serial and the parallel algorithm separately the following sentence which consumed the same CPU idle operation：

```
For (int tempI = 0; tempI < N; tempI ++)
        temp = temp*2 + tempI;
```

(Variable N meaning: CPU idle operation weight)

B. Experimental Environment and Test Result

Experimental environment one:

CPU integer: 1; Machine type: X86 Family 6 model 8; Basic frequency: 800MHZ; Operating system: Windows 2000; Programming language: C # [7].

Experimental environment two:

CPU integer: 4; Machine type: HP server; Basic frequency: 800MHZ; Operating system: Windows 2000; Programming language: C # .

Under above two kind of experimental environment, the algorithm all can discover Fig. 1 the most short-path .Its algorithm execution time and ratio (when $N=10^X$), as shown in Table I:

TABLE I
DIJKSTRA ALGORITHM EXECUTION TIME AND RATIO

		X=6	X=7	X=8	X=9
SP	S (ms)	47	340.3	3142	32254
	P (ms)	84.7	495.6	3307.3	33951.5
	R	0.55	0.86	0.95	0.95
MM	S (ms)	49	391.2	3822.3	37947.61
	P (ms)	33.11	106.59	987.67	9582.7
	R	1.48	3.67	3.78	3.96

SP= Single processor; MM= Multiprocessing machine;
S= Serial; P= Parallel; R= Ratio; ms= Millisecond.

V. Conclusion

Is not rare by above experimental result the following conclusion: Even if the ordinary serial procedure operating in the multi-processor system is also unable to use the corresponding hardware resources fully, with little difference of on single plane system; But use multithreading technology development parallel application procedure, when it operates the multiprocessing system, along with each thread computation load enlarging (in the above experiment is, its performance which increases through computation weight manifests) it has the distinct enhancement, and then the performance enhances the scope and the processor integer is proportional. The fact had further proven above theoretical analysis is correct. The parallel programming environment and the tool always lack the long vegetal period, and it is unable to renew through the unceasing edition, form with nowadays PC machine or the workstation equally easy to use environment and the tool. When carrying on the parallel programming using the multi-thread technology, it does not need again to consider the concrete hardware factor[6]. Now the majority operating system generally supports the multi-thread technology, in addition C # language its good cross platform[7], thus it causes when carrying on the parallel programming, it only need pay attention with question itself which remains unresolved. Thus it said the multi-thread technology for the design and the realization of the parallel algorithm open one new way.

References

[1] Guo-liang Chen. Parallel algorithm design and analysis. Beijing: Higher education publishing house, 2002, 433~434.
[2] Wei-min Yan, Wei-min Wu. Construction of data. Beijing: Qinghua University publishing house, 2001.
[3] Yijie Han, V.Y.Pan,J. H. Reif, Efficient Parallel Algorithms for Computing All Pair Shortest Paths in Directed Graphs, Algorithmica (1997) 17: 399–415.
[4] Hong,Cheng An, Guo-Liang Chen. Parallel programming model and language. Journal of Software, 2002.
[5] Jin-long Ji, Jin-li Zhong. Parallel programming model, language and translation technology. Small microcomputer system, 1995.
[6] Dai-ping Ji, Shou-wen Luo, Xin-yi Zhang. The parallel procedure develops the platform architecture the formalized research. Computer applied research, 2004.
[7] H.M.Deitel, Hao-han Ge,Yong-tao Tang. C # university course. Qinghua University publishing house, 2003.

Changle Lu received Bachelor degree in Computer application technology in Yantai Normal University in China. Now he is studying towards Master degree in Computer application technology in Tianjin Polytechnic University in China. His interest is in the field of algorithm design. Now he is making his graduation topic of realizing high performance video frequency encoder throw FPGA.

Yong Chen is a associate professor of Computer Technology and Automation of Tianjin Polytechnic University. His research direction is parallel computation and computer architecture (phone: +86-13752679603, 022-24585035; e-mail:luchangle_1815@163.com).