doi:10.3772/j.issn.1006-6748.2014.04.009

Parallel scheduling strategy of web-based spatial computing tasks in multi-core environment[①]

Guo Mingqiang (郭明强)[②] [*], Huang Ying[**], Xie Zhong[*] [**]
([*] Faculty of Information & Engineering, China University of Geosciences, Wuhan 430074, P. R. China)
([**] GIS Software and Application Project Research Center of the Educational Department, Wuhan 430074, P. R. China)

Abstract

In order to improve the concurrent access performance of the web-based spatial computing system in cluster, a parallel scheduling strategy based on the multi-core environment is proposed, which includes two levels of parallel processing mechanisms. One is that it can evenly allocate tasks to each server node in the cluster and the other is that it can implement the load balancing inside a server node. Based on the strategy, a new web-based spatial computing model is designed in this paper, in which, a task response ratio calculation method, a request queue buffer mechanism and a thread scheduling strategy are focused on. Experimental results show that the new model can fully use the multi-core computing advantage of each server node in the concurrent access environment and improve the average hits per second, average I/O Hits, CPU utilization and throughput. Using speed-up ratio to analyze the traditional model and the new one, the result shows that the new model has the best performance. The performance of the multi-core server nodes in the cluster is optimized; the resource utilization and the parallel processing capabilities are enhanced. The more CPU cores you have, the higher parallel processing capabilities will be obtained.

Key words: parallel scheduling strategy, the web-based spatial computing model, multi-core environment, load balancing

0 Introduction

As more and more users access geospatial information via the Internet, how to make so many concurrent users obtain requested information in the shortest possible time has become an urgent issue. The system response time is the important factor affecting the performance of web-based spatial computing system[1]. It mainly includes the following four aspects:

(1) Transmission time of request: It's the communication time from the web browser to the spatial computing server.

(2) Waiting time of request: It refers to the waiting time for processing requests in the spatial computing server.

(3) Processing time of geospatial information: It's the time of dynamically generating the results of the requests in the spatial computing server.

(4) Querying time of geospatial information: It refers to the time of querying the geospatial data from the background geographic database[2].

If we want to improve the performance of spatial computing server, it must meet two conditions. One is to support a large amount of users' concurrent access. The other is to shorten the average response time of requests to the maximum extent. The existing web-based spatial computing systems mainly use hardware or software methods to achieve this goal. By hardware, using a dedicated load balancer, improving the processing speed of CPU and increasing memory capacity of spatial computing server are the common methods. These methods have disadvantages such as high cost, poor scalability and so on[3]. Cheap and effective load balancing mechanisms are provided by software. Refs[3-11] have studied load balancing algorithms to some extent. The basic idea of these algorithms is using some strategies to allocate concurrent access requests to each server in the cluster. It can enhance the parallel processing capability of the entire spatial computing system to meet the needs of a large amount of concurrent access. But these algorithms only take into account

① Supported by the China Postdoctoral Science Foundation (No. 2014M552115), the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan) (No. CUGL140833) and the National Key Technology Support Program of China (No. 2011BAH06B04).

② To whom correspondence should be addressed. E-mail: 8156485@qq.com

Received on Oct. 10, 2013

how to balance the load from the requests to each server in the cluster, they do not consider how to improve the parallel processing capability inside a single server.

Today, the computer's processor has been experienced from single-core, high-frequency CPU to multi-core one. Multi-core computing platform provides a new method to improve the concurrent performance of servers[12]. Relying on the advantages of high-performance, low power consumption and low cost, multi-core CPU gradually dominates the market and has become the mainstream configuration of computer. Therefore, based on the multi-core environment, optimizing the parallel scheduling strategy of web-based spatial computing system to improve the concurrent processing capability of each server in the cluster will become a shortcut and lower-cost way.

In this paper, a parallel scheduling strategy of web-based spatial computing tasks in multi-core environment is proposed. It focuses on two issues. The first one is how to realize load balancing among different server nodes in the cluster environment. The other is how to make full use of the multi-core inside a single server node to improve the parallel processing capability of each server node. It ultimately achieves two levels of parallel task processing and realizes the load balancing among all servers and maximizes resources utilization of each server node. The simulation results show that the algorithm can effectively improve the CPU utilization, average I/O hits, network throughput, hits per second, and shorten the average response time of the web-based spatial computing system.

1 Parallel scheduling strategy of web-based spatial computing tasks

1.1 Web-based spatial computing model in multi-core environment

In order to achieve optimization of the performance of web-based spatial computing system in multi-core environment, a model is proposed as shown in Fig. 1.

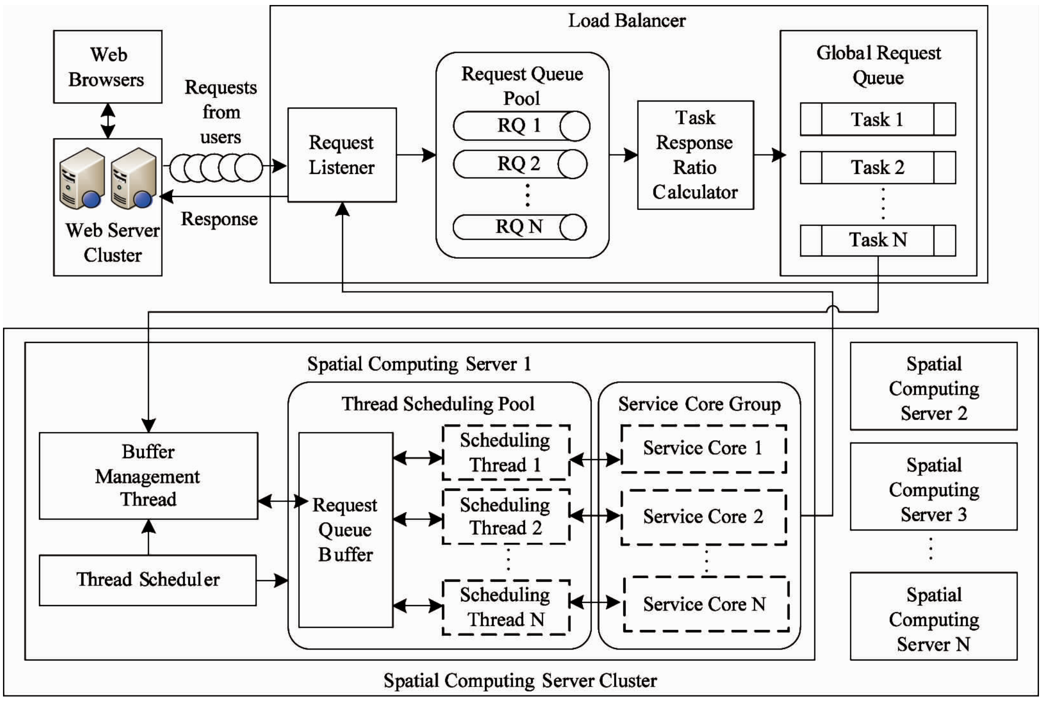

Fig. 1 The web-based spatial computing model in multi-core environment

The basic idea of the new model is as follows: (1) When users access the geospatial information, the concurrent access requests will be submitted to a web server of web server cluster. The web server will analyze parameters and submit these requests to the load balancer. (2) In the load balancer, the request listener will receive these requests and store them in the request queue pool. The task response ratio calculator will calculate the response ratio of each request by timer and then add them into the global request queue ordered by priority. (3) Each server node in the spatial computing server cluster will select the highest priority task from the global request queue. (4) The thread scheduler of each spatial computing server will schedule the buffer management thread to fill the request queue buffer which is in the thread scheduling pool. (5) The scheduling thread in the thread scheduling pool will get the tasks from the request queue buffer and assign them to the corresponding service core of the service core group. The scheduling thread and the service core will run in parallel. (6) The spatial computing server completes the processing of requests and returns the results to the load balancer. (7) Finally, the load balancer returns these results to the client. When concurrent users are increased, it only needs to add the spatial computing server nodes into the server cluster.

The model has two characteristics. One is the two-level parallel processing mechanism which is used among server nodes and inside one server node. It takes full advantage of multi-core parallel computing resources of each server node and realizes the task parallel scheduling and load balancing. Another is that the model changes the traditional passive load balancing methods. A new active task acquisition method is designed to implement the concurrent tasks allocation and load balancing in the cluster environment.

1.2 Task response ratio computing

Assuming t_s is the request submission time, t_b is the request start time, t_c is the completion time, t_p is the standard execution time, and T is the average response time of all concurrent requests, if only the request waiting time $t_b - t_s$ is considered in the queue and not t_p, it will lead to the delay of requests which have small t_p and long waiting time. The strategy in this paper considers the waiting time and execution time of requests at the same time and minimizes T. The request which has long waiting time and short execution time will have the high priority to be executed. The formula of T is given in Eq. (1). n is the total number of requests.

$$T = \frac{1}{n} \sum_{i=0}^{n} (t_c - t_s) \tag{1}$$

Now, the standard execution time t_p is calculated: the graphical display is taken for example. A bounding box of spatial data is used to measure the size of the requested content and calculate t_p for each request in the request queue pool.

The task response ratio calculator calculates the response priority of each request in the request queue pool. Assuming t is the current time, the task response ratio R is given in

$$R = 1 + \frac{(t - t_s)}{t_p} \tag{2}$$

Finally, based on R, the requests will be sorted and stored into the global request queue in the load balancer.

1.3 Multi-thread scheduling mechanism

The existing load balancing mechanisms of web-based spatial computing systems generally use the same following method: through a certain strategy, the load balancer calculates the load weights of each server in the cluster and orderly allocates requests to these servers. Servers accept requests passively and cannot process the requests in parallel immediately, which will lead to the delay of request processing and finally reduce the performance of system. Because the load balancer calculates the load weights at regular time but not in real time, it will lead to inaccurate results and unfair task assignment.

In the new model designed in this paper, it is not needed to calculate the load weight. A request queue buffer for each server node is designed and the buffer management thread is used to manage this buffer. The server node in the cluster can get the requests actively. The working principle of the thread is shown in Fig. 2. The goal is to allow all scheduling threads in the thread scheduling pool to access the request queue buffer concurrently and avoid task queuing problem caused by the lock operation when the spatial computing server access the global request queue directly.

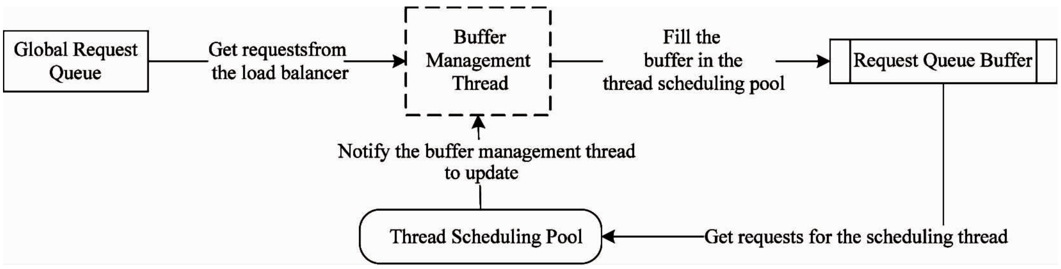
Global Request Queue
Get requestsfrom the load balancer
Buffer Management Thread
Fill the buffer in the thread scheduling pool
Request Queue Buffer
Notify the buffer management thread to update
Thread Scheduling Pool
Get requests for the scheduling thread

Fig. 2 The working principle of dedicated thread of request queue buffer

All scheduling threads in the spatial computing server run in parallel. Each scheduling thread accesses the request queue buffer concurrently to get the highest priority request and allocate it to the service core and notify the buffer management thread to update the request queue buffer at the same time. The buffer management thread gets new requests from the global request queue to fill the request queue buffer and ensure the number of requests in the buffer equals to the number of scheduling threads in the thread scheduling pool. Thread scheduling process is shown in Fig. 3.

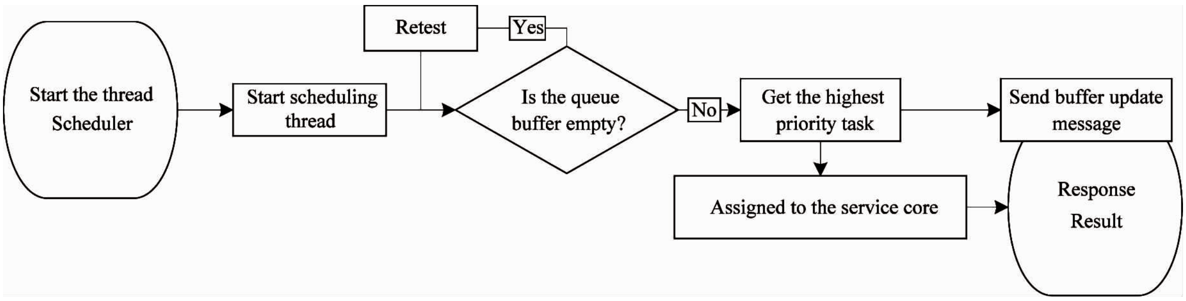
Start the thread Scheduler
Start scheduling thread
Retest
Yes
Is the queue buffer empty?
No
Get the highest priority task
Send buffer update message
Assigned to the service core
Response Result

Fig. 3 The flow chart of thread scheduling process

1.4 Model performance analysis

There are a lot of evaluation standards of computing model for parallel processing capability. The most commonly one is to calculate speed-up ratio. The speed-up ratio is used to reflect the actual effectiveness that the model can achieve. It is the most traditional evaluation standard for parallel algorithm in the multi-core cluster environment. It can be represented as

$$\psi(n,p) \leqslant \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n)/p + K(n,p)} \quad (3)$$

In Eq. (3), $\psi(n, p)$ is used to represent the speed-up ratio when we solve the problem of size n on p processors. In $\psi(n,p)$, the serial tasks are represented by $\sigma(n)$; the parallel tasks are represented by $\varphi(n)$; $K(n,p)$ is the computational overhead for executing requests in parallel.

According to Eq. (3), the Amdahl's theorem can be derived as

$$\psi \leqslant \frac{1}{f + (1-f)/p + K(n,p)} \quad (4)$$

In the Amdahl's theorem, ψ is the maximum speed-up ratio of parallel computing, f is the proportion of serial computing part in the whole computing tasks.

The Amdahl's theorem provides a method to predict the upper limit of speed-up ratio when a certain number of processors are used to solve problems in parallel. It can help us to determine whether an algorithm needs parallel optimization.

In the web-based spatial computing model proposed in this paper, it is assumed that T_s is the serial computing tasks and T_p is the parallel computing tasks. T_c is the whole computing tasks including T_s and T_p. The proportion of serial computing tasks f is given in

$$f = \frac{T_s}{T_c} = \frac{T_s}{T_s + T_p} \quad (5)$$

The model is analyzed by Microsoft Visual Studio 2010 performance analysis tools. Using the CPU sampling analysis tool and Eq. (5), a result is got that 5 percent of tasks in the model needs to be executed serially. Two servers are used to deploy a cluster and each server has 8 CPU cores. According to the Amdahl's theorem, the maximum theoretical speed-up ratio of the traditional model can be calculated which uses the existing load balancing mechanisms. The formula is

$$\psi_1 = \frac{1}{0.05 + (1-0.05)/2} \approx 1.9 \quad (6)$$

In the new strategy proposed in this paper, two level mechanisms are used to optimize the model performance. In Fig. 1, the service core group of each spatial computing server node in the cluster can run in parallel under the scheduling of the thread scheduling pool. The same hardware deployment is used through the CPU sampling analysis tool, 26 percent of the task needs serial execution. Because two server nodes are used, the first parallel part is 0.26/2. Besides, because each server node has eight CPU cores, the second parallel part is (1 − 0.26)/16 = 0.74/16. According to the Amdahl's theorem, the maximum theoretical speed-up ratio is calculated as

$$\psi_2 = \frac{1}{0.05 + (1-0.05)\cdot\left(\frac{0.26}{2} + \frac{0.74}{16}\right)} \approx 4.59 \quad (7)$$

From Eq. (6) and Eq. (7), the maximum speed-up ratio of the new model is about 1.41 times higher than the traditional model. It shows that the new strategy can obviously optimize the performance of the web-based spatial computing model.

2 Experiment and results analysis

2.1 Experimental parameters

The IBM blade server center in high speed intranet is used to build the test environment. The national geological map data is used to do the simulation experiment to verify the performance of the new model designed in this paper. Finally the comparison of the traditional model and the new one is given. The network topology structure of experimental environment is shown in Fig. 4, the system simulation parameters are shown in Table 1.

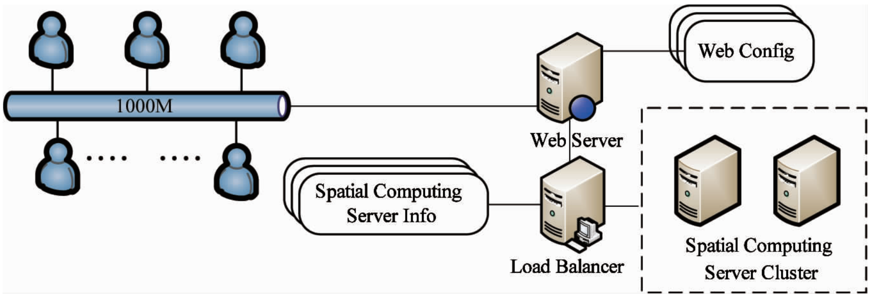
1000M
Web Config
Web Server
Spatial Computing Server Info
Load Balancer
Spatial Computing Server Cluster

Fig. 4 The network topology of simulation

Table 1 Simulation parameters of system

Simulation Parameter	Parameters content
Client	200 clients in Gigabit Ethernet, 64 ports Ethernet Switch
Web Server	Sufficient processing capability
Load Balancer	One server, CPU: Intel® Xeon® E5620 2.40 GHz (8cores), memory: 8GB
Spatial Calculation Server Cluster	Two servers, CPU: Intel® Xeon® E5620 2.40 GHz (8cores), memory: 8GB
Test data volume	12.27GB
Simulation time	5h
Testing software	HP LoadRunner 11
Statistical parameter	200 concurrent users. collect CPU Utilization, Hits per Second, Average IO Hits, Average Throughput and Average Response Time

2.2 Test result analysis

2.2.1 Analysis of result data

Table 2 shows the test result of the computing servers collected by the LoadRunner software in various test scenarios. The result data contains CPU utilization, hits per second, average IO hits, average throughput and average response time.

Table 2 The results data of stress test

Threads number	1	2	4	8	16
CPU Utilization(%)	24.13	38.05	50.95	57.15	57.5
Hits per Second /s	14.917	24.625	38.191	47.789	50.356
Average IO Hits /s	15.036	28.873	38.572	48.295	51.081
Average Throughput (M)/s	1.615	2.935	4.526	5.672	6.081
Average Response Time (s)	3.432	2.104	1.603	1.447	1.527

The test server's CPU has eight cores. According to the test results shown in Table 2, the following conclusions are got:

(1) When the number of threads is less than eight, some CPU cores are idle. With the increase of thread number, more and more CPU resources can be used. The throughput will grow linearly and the average response time will reduce in the same way. Finally, it achieves the performance improvement.

(2) When the number of threads is eight, we can make full use of all the CPU cores. Each service core run in parallel, parallel processing performance will achieve the optimal state.

(3) When the number of threads is more than eight, CPU will do additional thread switching operation frequently, so the parallel processing capability can not increase linearly.

(4) From the result data (especially the average response time) in Table 2, we can infer that the new model's parallel processing performance is increasing linearly, which demonstrates that the task allocation strategy can balance the load among all the processors.

2.2.2 Parallel efficiency and the speed-up ratio analysis

The average response time collected by LoadRunner software will directly reflect the parallel processing capability of the computing server. According to the average response time of the results data in Table 2, the speed-up ratio of the optimized model designed in this paper can be calculated. Fig. 5 shows the parallel efficiency upgrade time of the new model when different number of threads are used.

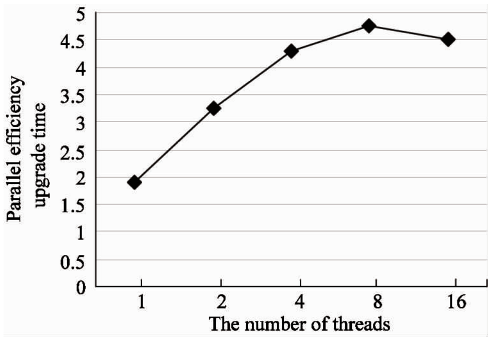

Fig. 5 Parallel efficiency upgrade time

Fig. 5 shows that the speed-up ratio achieves the maximum value 4.74 when eight threads are used. It is consistent with the theoretical analysis of speed-up ratio in Section 1.4. Since the server's CPU has only eight cores, when the number of threads is less than or equal to eight, parallel efficiency and speed-up ratio is proportional to the number of threads, which has linear speed-up ratio and maintains a better parallel efficiency. The linear curves also demonstrate that the new model can balance the load among multiple processors. When thread number is more than 16, CPU has to do additional thread context switching, and each thread competes the limited resource, so the performance can not be improved but fallen.

3 Conclusions

A parallel scheduling strategy of web-based spatial computing tasks in multi-core environment is studied and a new model is designed based on the strategy. In the model, two level parallel processing mechanisms are used to make full use of the multi-core resources of servers in cluster. In order to verify the performance of the model, a performance test is done between the traditional web-based spatial computing model and the new one. The experimental results show that the optimized model can improve the CPU utilization, I/O throughput, and hits per second. When it's confronted with large concurrent access, it can shorten the average response time greatly and give a better experience to users. It shows that the new model has a better stability and capability of resistance to high load.

References

[1] Zhang S, Wu CM. Research progress on high performance spatial analysis of GIS. *Journal of Sanming University*, 2011, 28(6):24-25

[2] Xiao X Z, Su F Z, Du X Y, et al. Performance analysis and optimization of WebGIS platform. *Geomatics & Spatial Information Technology*, 2005,28(4):1-3

[3] Jiang F, Zhou B Q, Wang H. An effective loading-balancing framework for distributed WebGIS. *Microcomputer Information*, 2006, 22(10):215-218

[4] Gong Z, Tang W, Bennett D A, et al. Parallel agent-based simulation of individual-level spatial interactions within a multicore computing environment. *International Journal of Geographical Information Science*, 2013, 27(6): 1152-1170

[5] Cheng G, Jing N, Chen L. A theoretical approach to domain decomposition for parallelization of digital terrain analysis. *Annals of GIS*, 2013, 19(1): 45-52

[6] Li J, Jiang Y, Yang C, et al. Visualizing 3D/4D environmental data using many-core graphics processing units (GPUs) and multi-core central processing units (CPUs). *Computers & Geosciences*, 2013, 59:78-89

[7] Yavits L, Morad A, Ginosar R. The effect of communication and synchronization on Amdahl's law in multicore systems. *Parallel Computing*, 2014, 40(1):1-16

[8] Munir A, Ranka S, Gordon-Ross A. High-performance energy-efficient multicore embedded computing. *IEEE Transactions on Parallel and Distributed Systems*, 2012, 23(4): 684-700

[9] Huang Y, Xie Z, WU L, et al. A WebGIS model based on cluster scheduling load-balancing algorithm. *Earth Science (Journal of China University of Geosciences)*, 2010, 35(3):407-414

[10] Buchty R, Heuveline V, Karl W, et al. A survey on hardware-aware and heterogeneous computing on multicore processors and accelerators. *Concurrency and Computation: Practice and Experience*, 2012, 24(7): 663-675

[11] Zhu L, Shen W M, Li R. Genetic algorithm-based dynamic load balancing for server cluster in network GIS. *Geomatics and Information Science of Wuhan University*, 2011, 36(6):722-725

[12] Zhang S Q, Cheng G, Chen L. Research on parallel algorithm of edge extraction based on multi-processor. *Computer Science*, 2012,39(1):295-298

Guo Mingqiang, born in 1984. He received his Ph. D degree from China University of Geosciences in 2013. He also received his B. S. degree from China University of Geosciences in 2007. His research focuses on key techniques for WebGIS performance optimization.