Send Orders for Reprints to reprints@benthamscience.ae

The Study And Implementation of Lighting Parallelization Algorithm

Zhang Wei-wei * and Zhang Xin *

School of Electronic Engineering, Xi'an University of Posts and Telecommunication Xi'an, China

Abstract: According to slowly serial processing speed in the photorealistic rendering technology, the paper proposes many lights processing. Using parallel strategy of load balancing and optimizing calculation mode, ambient light 、 diffuse light and specular light are added when they are calculated separately. Different number of PE (processing unit) with different schemes is to improve the processing speed .The experimental results show that the parallelization of many lights can make a better use of resources and the processing power of multicore processors. The optimizing of the processing from serial computing into parallel computing accelerate the original operation, as well as resource utilization.

Keywords: Many lights, parallelization; sense of reality, phong model, photorealistic rendering technology.

1. INTRODUCTION

In OpenGL, light and objects in the scene can be calculated, which creates many different types of three-dimensional effects. As the light is indispensable in nature, drawing a realistic three-dimensional objects requires the disposing of the light. OpenGL lighting is an approximation realistic lighting. Attributes of light source in OpenGL include radiation, ambient, diffuse and specular light and so on. Material is an attribute of the object, indicating how the object is made of materials. Materials in OpenGL define the various light reflection rate of surfaces. The color ultimately reflected to the human eye by the objects in the scene is the color resulted from the multiplication of the RGB components of the light and the reflectance of the RGB components of the material. Currently used lighting algorithms in OpenGL include Gouraud model and Phong [1-2] model. In this paper, we focus on the parallelization [3-4] of Phong model algorithm.

2. VERTEX LIGHTING ALGORITHMS

A. Phong Model

Phong model is the one of the classical graphics lighting model, the model can be used to generate realistic graphics. Gouraud shading [5] using color interpolation methods to eliminate discontinuity between polygons. This can produce a smooth surface feel, each attribute of the vertex should be calculated in the stage of vertex shader, it has a relatively small amount of computation. Phong shading is similar to Gouraud shading, this method uses the color interpolation instead of vector interpolation. Results are more realistic. The work of calculation is pushed to the pixel shader stage with a relatively large amount of computation. Here is Phong lighting model algorithms [6].

B. Phong Shading Algorithm

Phong model supports several types of interactions between the material and light: ambient light, diffuse light and specular light, with considering the attenuation factor and the role of emitting light. For each color component, each point has several separate light sources, each component can be individually calculated. The vertex color is constituted of them [7, 8]. The following are separate descriptions:

- Ambient light: Ambient light is totally diffused in the environment, whose direction can't be identified. When the ambient light strikes a surface, it will be spread in all directions. The light is the ambient color according to the material properties of the environment after scaling values:

$$ambient_{light}\ ambient_{material} \tag{1}$$

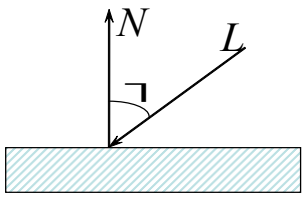

Fig. (1). Reflected ray

- Diffuse light: Diffuse light comes from a direction, the intensity of reflected light depend on not only the material but also the position of the light source .The position relative to the surface. If the light is irradiated to the front surface, it would be brighter. If the light sideways passing surface, it would be dark relatively. It is related to both the normal vector N and on the unit vector L between the position the light source and the vertex, just as shown in Fig. (**1**). Diffuse light component should consider whether the light irradiat-

1874-4443/15 2015 Bentham Open

ed on the vertex .At the same time ,the diffuse color of the light source and properties of the diffuse material:

$$(\max\{L \cdot n,0\}) * diffuse_{light} * diffuse_{material} \tag{2}$$

- Specular light [9]: Without considering the specular light, although scene images has effect of darkness and lightness, as well as three-dimensional effect, it seems that there is a lackness of the highlighted area is missing on the surface. Highlighted region comes from the reflection of shiny objects. The smoother the surface, the effect is more like a mirror reflection. Its value depends on the sum of these two unit vectors: unit vector $L=(L_X,,L_Y,,L_z)$ from the vertex to the position of the light source unit normal vetor $n=(n_X,n_Y,n_z)$of the vertex. At the same time, it is also related to the specular exponent shininess, specular color $pecular_{light}$ and $specular_{material}$ properties $specular_{material}$.

$$(\max\{s * n,0\})^{shininess} \times specular_{light} \times specular_{material} \tag{3}$$

- Attenuation factor: Taking into account the distance factor, the light source emitting from the light source to the surface, the light intensity gradually decreases with distance increases. Attenuation depends on the distance d between the light source position and the vertex, the constant attenuation factor, linear attenuation factor, the quadratic attenuation factor. While considering whether it is the spotlight light source, we need to determine whether the light is in the spotlight spotlights and whether the vertex is in the cone of light. If it is, then we should also consider unit vector from the vertex to the spotlight, the directiona of the spotlight, cut angle factor of spotlight. Attenuation factor and spotlight effects following formula:

$$attenuationfactor = \frac{1}{k_c + k_l d + k_q d^2} \tag{4}$$

$$spotlighteffects = (\max\{v \quad d,0\})^{GL\_SPOT\_EXPONENT} \tag{5}$$

- Light emitting of material: Emission light of the material simulates the light of an object, emission color of the surface can increase the strength of the object, its value is assigned to the RGB values of the parameter of *GL_EMISSION*..
- Complete lighting formula: In summary, the above results are superimposed, the vertex color [1]= material emission color + global ambient light (zoomed due to the material properties in the environment) +ambient light components, diffuse light components and the specular lights which are properly decayed (come from all the light sources). In the implementation of lighting calculations, the color values is within the range in [0,1], Light formula is showing as follows.

$$\begin{aligned} &emission_{material} \\ &+ambient_{light\,\mathrm{mod}\,el} \times ambient_{material} \\ &+\sum_{i=0}^{n\blacklozenge 1}\left(\frac{1}{k_c \pm k_l d \pm k_q d^2}\right)_i \times (spotlighteffects)_i \\ &\times\left[\begin{array}{l} ambient_{light} \times ambient_{material} \\ \pm(\max\{L\blacklozenge n,0\}) \times diffuse_{light} \times diffuse_{material} \\ +(\max\{s\blacklozenge n,0\})^{shininess} \times specular_{light} \times specular_{material} \end{array}\right]_i \end{aligned} \tag{6}$$

3. IMPLEMENTATION OF PARALLELIZED PHONG SHADING ALGORITHM

Parallel algorithms vary in features according to different categories of problem and architectures of parallel machines. A well-worked parallel algorithm is adaptive to the characteristics of the structure of parallel computer hardware system and the inherent parallelism of the problems to solve. For parallel computing, there is a very important principle, that is to manage to increase the proportion of computing time and reduce the frequency of communication. If the communication of processing units can coincide, the execution efficiency of the program will be promoted greatly.

The parallelism of shading algorithm can be embodied in two aspects—data parallelism and task parallelism[10-11], This paper puts forward a combined method with the strategy of both data and task parallelism, which is optimized in the pipelined design to achieve higher hardware utilization and decrease the intermediate data quantity. This parallel implementation is able to deal with data intensive tasks such as image processing and vertex shading, and make the system throughput speed up to a high level.

To accomplish the above calculation, the amount of computation is large. Traditional GPU using multiple vertex shading devices to do the calculation. The polymorphic array architecture processor (PAAG) is used to achieve parallel shading in this paper.

A. The Polymorphic Array Architecture Processor

As shown in Fig. (**2**), it is an efficient parallel processing array and structure which is suitable for graphic image The processing cell array structure is composed of multiple processor computer system clusters, each cluster is composed of 16 processing units with 4 * 4 array. The processing unit is a two-dimensional array composed of the PE. Each line has line controllers, each column also has column controllers. This makes such a machine can achieve highly efficient parallel computing by line level, data-level parallel computing and parallel computing operating level.

In order to validate the program, we use PAAG processor simulation system, compilation system, debugging environment and the parameter extraction tool for performance analysis. This is a complete set of software development tools providing a reliable guarantee for program execution and performance analysis.

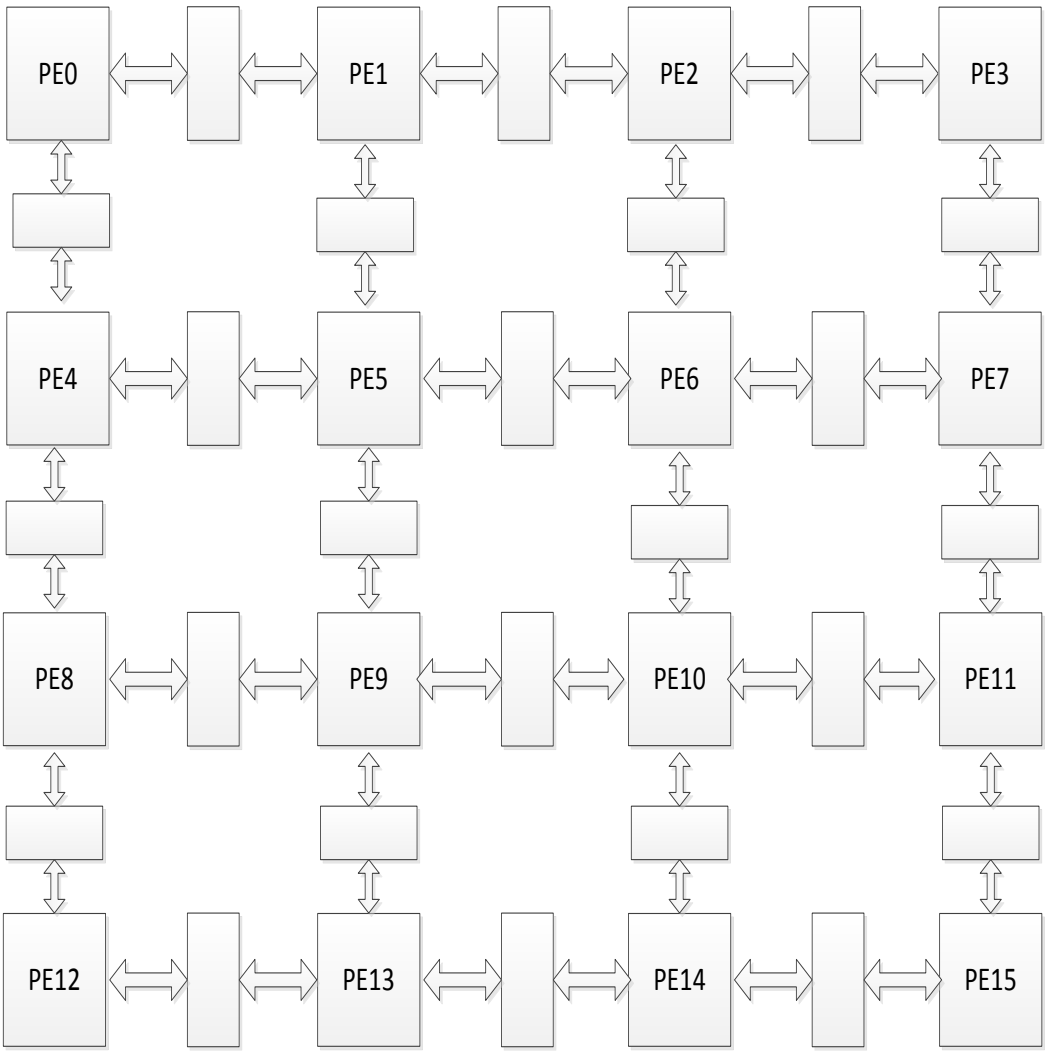

Fig. (2). The structure of the polymorphic array architecture processor.

Clock accurate simulation techniques are used in PAAG system, the individual levels of PE have carried out modeling, pre-processing module distributed the stored data to the various functional modules according to the connection between the modules and the communication protocols, processed data is sent into the post-processing module, which delay the data according to the internal software counter clock appropriate and then send the data to a pre-processing module, forming a loop, to achieve the communication between the various modules and the analog clock, in order to achieve the clock accuracy level modeling. The compilation system provides a compiler system from high-level language to the compiler, and assembler instructions from assembly instructions to machine readable machine code assembler, Its debugging environment used to validate the correctness of the prototype system, and performance analysis tools achieve speedup with collecting the corresponding parameter extraction algorithm program in a single PE and the running time in several PEs, what's more, it provides the basis for the optimization algorithm.

B Single Light Parallel Algorithm Implementation

1) Serial implementation: To get serial computation, vector standardization and the calculations of attenuation factor and spotlight effects should be achieved first. Then the calculations of diffuse light, specular light and ambient light component should be achieved. you can also change the order of them.the paper lists an order of the algorithm. Serial implementation is shown as Fig. (**3**):

2) Parallelization preliminary: Parallel processing of lighting algorithms requires the work to be seperated into discrete sections, which helps to solve several program instructions timely,at any time or at the same time ,which will shorten the calculation time. Parallel algorithm devids the algorithm into multiple parts due to the formula in the adding portion, ambient light, specular light, diffuse light and the attenuation factor would be calculated in every portion. Preliminary divided idea is shown as Fig. (**4**):

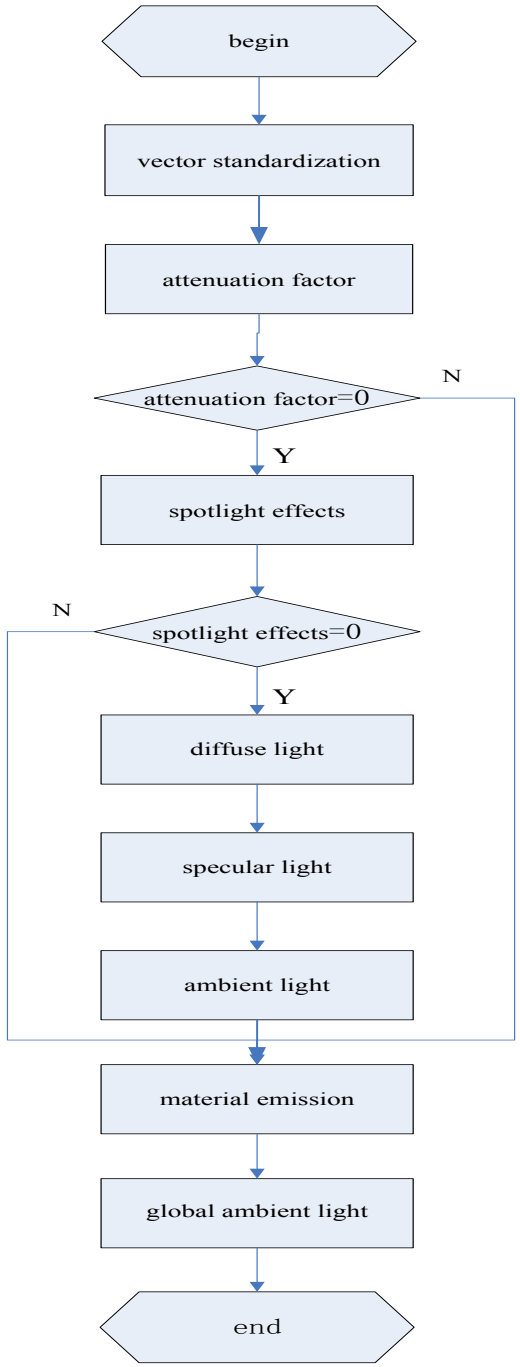

Fig. (3). Serial structure.

3) Parallelization implementation: Since the amount of the calculationg in each part could be different , the simple division will lead to load imbalance,which would affect efficiency of the algorithm, it uses a load more balanced parallel strategy, re-optimization tasks is shown in the following Fig. (**5**).The calculation of the vectors S1,S2(required by specular light) is put into another PE,the computing and the merging of emission light and the global light are put into the same PE, which ensurs the relevance of the data, Serial implementation is shown as follows:

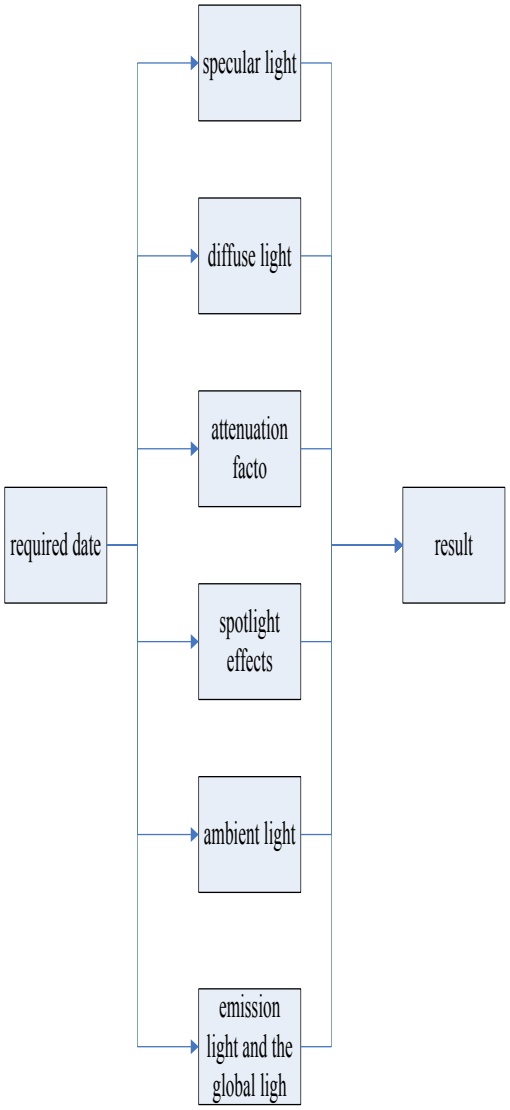

Fig. (4). Preliminary parallelization structure.

The implementation of parallelization algorithm: When we achieve the parallelization of The Phong shading algorithms, the source allocation and the full use of the source should be considered so that plura PE could be interdependent and synchronized. Specific implementation is shown below:

The implementation of the parallelization used seven PE to complete:

PE0: Normalized the vector from the vertex to spotlight, then send results into PE1.

PE1: Calculate the sum of the distance between the vertex and the observation point and the distance between the vertex and the light source in the specuar light. First, determine whether there is specular light, after that, receive calculated the value of the desired unit vector from PE1. Then, it is required to calculate again. What's more, the results should be normalized, calculate the required value of the specular light component, and finally send the result to PE2.

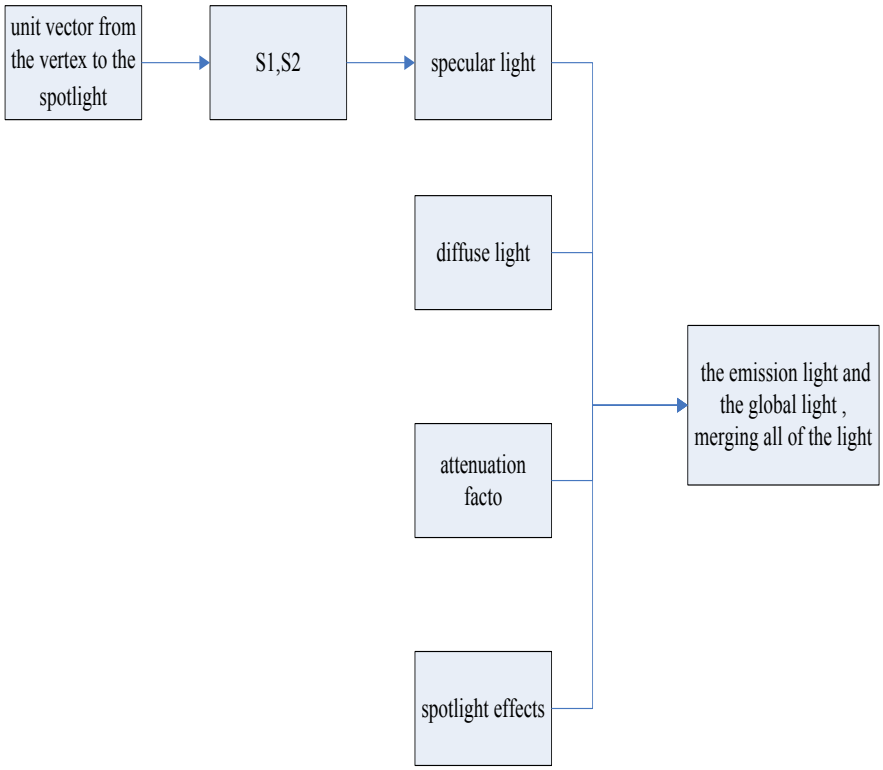

Fig. (5). parallelization structure

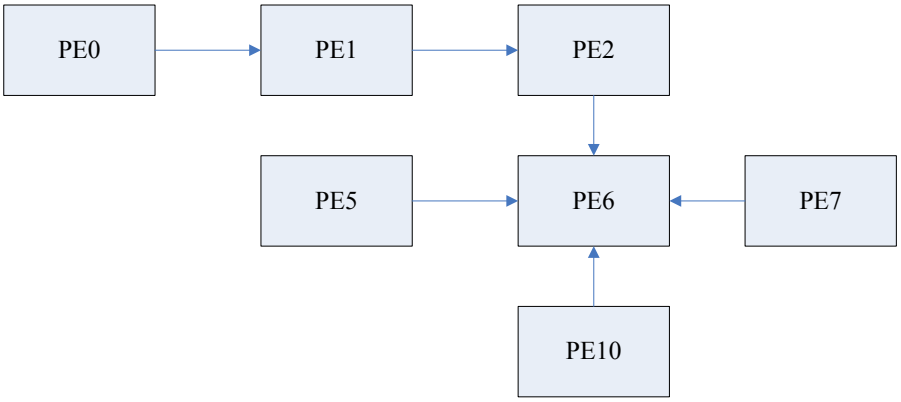

Fig. (6). The structure of the communication between PE

PE2: Do the specular light calculations. According to the calculation formula (3) of the specular light and the desired value obtained from PE1, calculated specular light ,send result to PE6.

PE5: Calculate the diffuse light. According to the formula (2) of the deffuse light , and the the results would be sent to PE6 from PE5.

PE7: Calculate the attenuation factor. According to the calculation formula (4) of the attenuation factor and the given values, we can get the attenuation factor.we should then determine whether the result is0. if it is, then only the material emitting light and the global ambient light could influence on the lighting effects , if it is not, The results will be fed into the PE6.

PE10: Calculate spotlight effects. First, determine whether the spotlight effect exists, if it exists, the spotlight effects are calculated according to the formula(6).Then we need to determine whether the result is 0.If it is 0, only the material emitting light and global ambient light influence on lighting effects, if it is not, results would be sent into PE6.

PE6: Calculate the ambient light and each of the above ingredients are combined,.Then add the material emitting light and global ambient light to get the final lighting color .

In summary, the final results are stored in PE6 and have achieved the lighting parallelization.

C Mang Lights Parallel Algorithm Implementation

If there are mang lights, 1 PE can run one light in parallel algorithms.

As shown in Fig. (**6**), each PE can calculate each light, the final results are added. According to eight lights, 1 PE

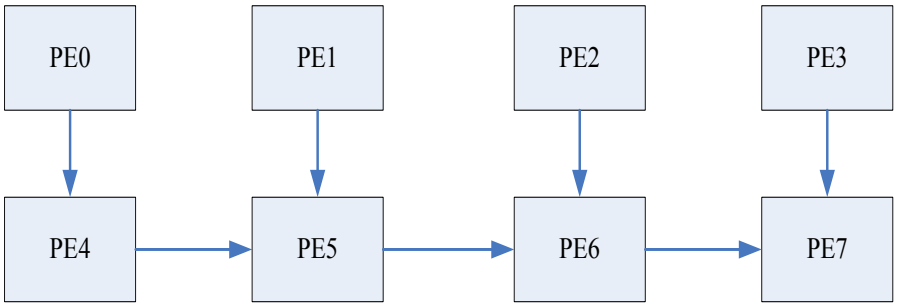

Fig. (7). Parallelization structure of 8 PE.

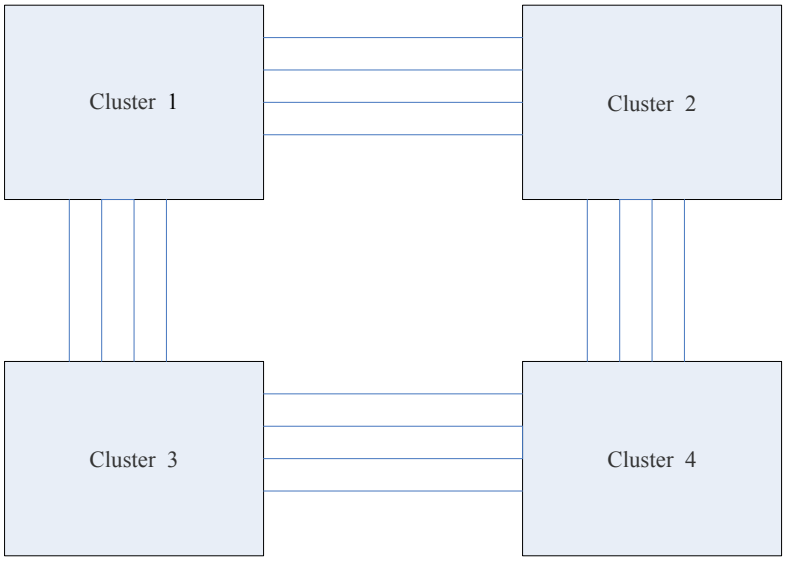

Fig. (8). The expansion of basic clusters.

runs one light. PE0 calculates the first light. PE1 calculates the second light. PE2 calculates the third light. PE3 calculates the fourth light. PE4 calculates the fifth light. PE5 calculates the sixth light. PE6 calculates the seventh light. PE7 calculates the eighth light. Finally, 8 PE run simultaneously, alll results are added.

D New Many Lights Parallel Algorithm Implementation

As it known, PAAG processor can be extended.It is able to form clusters of high-rise structures. There are 1024 processing units in all. Each row and column controller controller have its own program memory. The specific expansion mode is shown in Fig. (**7**):

RESULTS AND ANALYSIS

With Amdahl's law, the system performance of improvement can be intuitively reflected by the value. It defines a speedup achieved by acceleration measures.

Speedup reflects the rate between using accelerated measures to complete a task and not using accelerated measures to accomplish the same task to speed up. Therefore, the speedup can be defined as:

$$\text{Speed} = \frac{\text{The performance of achieving the whole task when using accelerated measures}}{\text{The performance of achieving the whole task without using accelerated measures}}$$

Can also be defined as :

$$\text{Speed} = \frac{\text{The performance of achieving the whole task without using accelerated measures}}{\text{The performance of achieving the whole task when using accelerated measures}}$$

It is convenient for us to estimate if the latter define is used. Using different PE, amount and different assignments of different ways to get a speedup, the results would also be different. Following is a serial processing program of the vertex and results running parallel program 7 PE:

From the Table **1** above, for a single light, using 7 PE parallel, speedup can reach 2.78.

From the Table **2** above, when there are eight lights, according to Fig. (**8**), speedup can reach 7.46.

Table 1. Parallelization results of 7 PE.

	Running Time (Clock)	Block Time (Clock)	Exec Time (Clock)	Speedup
PE0	199	0	199	2.87
PE1	204	148	352	
PE2	117	287	404	
PE5	323	0	323	
PE6	238	239	477	
PE7	307	0	307	
PE10	383	0	382	

Table 2. Parallelization results of 8 lights.

	Running Time (Clock)	Block Time (Clock)	Exec Time (Clock)	Speedup
PE0	1294	0	1294	7.46
PE1	1294	0	1294	
PE2	1294	0	1294	
PE3	1294	0	1294	
PE4	1310	0	1310	
PE5	1470	0	1470	
PE6	1325	162	1487	
PE7	1314	177	1491	

Table 3. Speed-up ratio of 4 clusters 8 lights.

Calculation Method	Speedup
Single lamp parallel	2.87
Original eight parallel light	7.46
New presents eight parallel light	23.43

As can be seen from Table **3**, when only one light, speedup can only reach 2.87. It shows that small-scale parallelism is not dominant. According to the original method of calculation, eight lights parallel speedup achieves up to 7.46. However, using the method of figure 8,the speedup can reach 23.43. With the increasing scale, the time has obvious differences. It indicates that large-scale computing parallelization effect is obvious.

CONCLUSION

This paper describes the light parallel algorithm on multicore processors. It specifically describes distribution scheme of parallel algorithms and evaluates the indicators of parallelization effect. For more lights, different parallel structures have different speedup. The use of the proposed scheme, speedup improves a lot.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGEMENTS

This work is supported the Education Science fund of the Education Department of Shaanxi, China (No. 14KJ1672), and Xi'an University of Posts and telecommunications in youth fund (ZL2014-14)

REFERENCES

[1] Dave Shreiner, The Khronos OpenGL ARB Working Group, OpenGL Programming Guide, China Machine Press, 2010.

[2] E. Angel, "Interactive Computer Graphics," Tsinghua University Press, 2012.

[3] P.Pacheco, "An Introduction to Parallel Programming," China Machine Press, 2011.

[4] D. Bo, "The Research of Parallel Algorithm And Applicationg," Chengdo: University of Electronic Science And Technology of China, pp.55-57, 2012.

[5] H. Li, and L. Wang, "Parallel programming languages on multi-core and many-core architectures," *Information Technology Letter*, vol. 10, no.1, pp.23-30, 2012.

[6] X. Chao, and D. M. Lian, "Research and Analysis of Parallel Computing System Speedup," *Computer Engineering and Applications*, vol. 12, no.6, pp. 66-68, 2013.

[7] T. Li, and L. Xiao, "Implementation of paralled biological computing based on polymorphous parallel,processor," *Journal of Xi'an University of Posts and Telecommunications*, vol. 17, no.3, pp. 41-47, 2012.

[8] S. Kang, J. Danf, and S. Zhonf, "Paralled algorithm of image homumorpic filtering baded on SMD PE array," *Journal of Xi'an Polytechnic University*, vol. 24, no.3, pp. 302-305, 2010.

[9] H. Liao, T. Inomata, I. Sakuma, and T. Dohi, "3D augmented reality for MRI-guided surgery using integral videography autostereoscopic-image overlay," *IEEE Transactions on Biomedical Engineering* ,vol. 57,no.6, pp.854-857, 2010.

[10] H. Li, and L. Wang, "Parallel programming languages on multi-core and many-core architectures," *Information Technology Letter*, vol. 10, no.1, pp.23-30, 2012.

[11] T. Li, "A Polymorphic Array Architecture for Graphics and Image Processing", 2012 5th Int. Symp. Parallel Architectures, Algorithms and Programming, Taipei, *IEEE Computer Society CPS,* pp. 242-249, 2013.

Received: June 16, 2015 Revised: July 12, 2015 Accepted: September 11, 2015