International Journal of Hybrid Information Technology
Vol.8, No.2 (2015), pp.121-134
http://dx.doi.org/10.14257/ijhit.2015.8.2.11

Design and Application of Small Scale Cluster System Based on OpenFOAM

Yiran Wang and Wengang Zhou

School of Computer Science and Technology, Zhoukou Normal University, Zhoukou, 466001, China
wangyiran76@163.com

Abstract

In order to solve the simulation problem of steady compressible turbulent combustion, a small scale high performance parallel cluster system was designed. It based on the open source CFD platform OpenFOAM, and was used to to simulate the bluff－body stabilized Sydney flame hm1 which refers to the detailed chemical kinetic mechanisms of coal gas and air combustion reaction. The flow of combustion reaction component concentration distribution and temperature curve diagram are plotted according to the computational results. The comparison between the simulation and the experimental results shows that this parallel cluster system has efficient computation for steady compressible turbulent combustion flow.

Keywords: *Turbulent combustion, Numerical simulation, Cluster system, Performance analysis, .k-ε two-equation*

1. Introduction

As the development of computer disposal ability, the performance of a single personal computer becomes more and more high. But it can't deal with some large scale matters which commonly exist engineering calculation. It need make many computers work together to settle correlative problems. So the real application introduces cluster technology. The cluster is an aggregation containing a set of unattached computers which are linked by high-powered network. Every computer can be regarded as a node which not only be as a single resource for interactive users, but also can works systematically for parallel computation. The cluster is a kind of parallel computer system architecture which owns low-cost, apt to construction and good expansibility characters [1]. Computational Fluid Dynamics (abbreviation CFD) [2], which widely be used in many areas, touches on calculations that are enormous size, and notoriously slow. During the requirement increase of all kinds of CFD software network application, how to realize the coupling high performance computing of numerical simulation system is the problem needed to be solved quickly at present. The paper probes into the feasibility of the numerical simulation of combustion flows field on a small scale cluster system, and of the parallel computation of Open Field Operation and Manipulation platform.

2. Design of Parallel Computation Platform

CFD is the combination of the modern hydromechanics, numerical mathematics and computer science, and is a borderline subject which has strong vitality. It uses computer and a variety of discrete numerical calculation methods to make experiments, simulate with computer, research and analyze on problems of CFD so as to resolve practice questions, uncover new physical phenomenon and open new research realms [3]. There are many complicated mathematics problems and calculations such as the solving of application serial computation, especially the large amount grids calculation [4], which don't meet the demand of the real project's need. The parallel computation of cluster system [5]makes so many intricate problems numerical simulation of CFD become efficient. The deployment of cluster system comprises software configuration and hardware configuration.

2.1. Software Configuration

Every computer node has its own processor, EMS memory and operating independently OS in platform of cluster system. It can't directly visit processors, EMS memories and other devices of other nodes. But cluster environments request these accessing, it needs the support of inner network and protocols which can provide some important functions. These functions include that every computer node can visit each other, use the same computational programming and make corresponding initialization, and offer high efficient user identity authentication and execute the calculation instructions. In order to realize these functions, NFS server and SSH server becomes the basic network devices in cluster environments. NFS server [6] is a used distributed file system, which makes remote client machines perform the file access operations. When parallel task starts up, it can make all computer nodes of cluster system load the executable codes of parallel programming, and share the input and output data files by using NFS. NFS is a credible protocol which doesn't preserve user information in the process of services, and the problems of client does not influence server. Although NFS exits some security menace, these problems are not obvious in inner local area network.

2.2. SSH Server

Secure Shell transfers data by encryption method of SSL to avoid the feasibility of data being intercepted and changed [7]. During configured SSH server in platform environment of cluster system, it can make highly efficient user authentication by reliable IP examination for computer nodes. For cluster system, the job is to do parallel computation. If no password logons, it will bring more high calculation efficiency unquestionably. Parallel programming language based on message passing parallel programming model contains MPI and PVM. MPI [8]is now the most popular distributing storage parallel programming environment. It not only owns good transplantation, powerful function and high efficiency, but also has many different free of charge application editions which includes MPICH, LAM, CHIMP and MPI/PRO. Almost all computer manufacturers provide the support for it, which is incommensurable for other parallel environments. MPI absorbs the advantages of many existent systems, becomes a standard of message transfer library which are more popular and used in parallel programming. MPI could be installed in many platforms such as Windows NT/2000, Linux, and Unix. The library function can be embedded in C, C++ and Fortran programming languages. MPI has many methods to realize its functions, in the paper we adopts OpenMPI [9] which is an open source platform.

2.3. Hardware Configuration

To set up small scale cluster system comprises eight personal computers, and every machine has the same hardware configuration. The processor is INTEL P4 2.0G, the capability of EMS memory 512MB, 80G hard disk, and 100M b/s network card. The cluster system makes use of twisted-pair to take eight PC form a local area network by Ethernet switches. Among these nodes, one is the master node which is in charge of task assignment and process management, the others are slave nodes which participate in computation.

That cluster system implements parallel calculation of CFD needs to install related software. Here we use OpenFOAMas the calculation platform which is more popular in recent years. The full name of OpenFOAM is Open Field Operation and Manipulation, its core is the class libraries of CFD which are compiled with C++ programming language and has efficient resolving model of partial differential equation. The source code of OpenFOAM is open for public, which is convenient for user to compile solver. OpenFOAM software can simulate complicated fluid flow, chemical reaction, conduct heat etc., and also can make numerical imitation for the analysis of structure dynamics, electromagnetic field. From the angle of program realization function, OpenFOAM comprises three models: core solver, pre-processing and post-processing [10]. Pre-processing mainly uses tools of gird design to work, like BlockMesh owned by OpenFOAM, also uses other commercial grid design software such as GridGen. In the paper, firebox grid design adopts GridGen, then transforms it into grid files which can be discriminated by OpenFOAM. The flow models supported by solver are Laminar, Reynolds time-averaged control equations (abbreviation RAS), large eddy simulation (abbreviation LES), and direct numerical simulation (abbreviation DNS). Post-processing mainly turns the result into visual image, like nephogram, combustion isosurface graphics. Modules of OpenFOAM composition are like the following Fig.1. One of primary function is the extremely efficient parallel computation. The function depends on extension program based on MPI contained the packages of third party. After software configuration on the cluster system, OpenFOAM should be installed on the master node, and other slave nodes can use the software from mirror images.

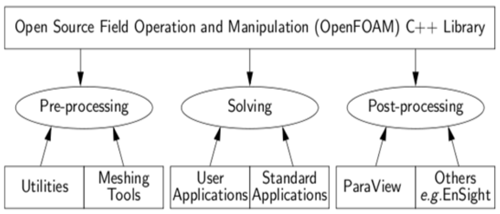

Figure 1. Overview of OpenFOAM Structure

3. The Performance Analysis of Cluster System

Linpack [11] is a linear algebra package which is compiled with Fortran language. It mainly be used in solving linear equation and Least-Square problems. The package offers all kinds of solving approaches of linear system, such as matrix operation. Linpack is the most popular test benchmark of computer calculation performance in the world. It appraises the floating-point operation capability of high-performance computer system during the test of solving ability of dense linear algebraic equations. The test results use float-point operation times every second (Flop/S) to express. Linpack test standard [12] comprises three different contents. One is test of n=100. The second is peak value performance oriented TTP test. The third is high parallel computation test which has no limits for array size. HPL (High performance Linpack) is the first normal public edition, parallel Linpack, widely used in Top500 test. HPL is a high performance test package of Linpack which is developed by A. Petitett and R. C. Whaley. HPL adopts C language and MPI to compile. The main arithmetic is row principal component block solving for large denseness linear equations, which is fit for MPP system and cluster. It has good performance and high efficiency. We think it is the best performance test program of Linpack at present. HPL test is mainly designed for the distributed storage large scale parallel computation system. User can set up any size of matter space, make use of any number of CPU, take advantage of Gaussian elimination optimum method to get the optimal test results.

HPL test program needs the support of MPI communication library environment and Basic Linear Algebra Subroutines (abbreviation BLAS) or Vector Signal Image processing Library (abbreviation VSIPL). BLAS library supplies high efficient subroutines about basic operations of vector and matrix, and MPI affords message transfer criterions which is independence of calculation platform. BLAS is used on some simple vector operation, for example, enhanced times for appointed vector and added it to another vector. Plentiful floating-point calculations in LINPACK algorithm are realized by calling BLAS. This provides conditions for algorithm which uses specific hardware but don't modify bottom. BLAS contains three layers [13], the first layer (the bottom) realizes the operation for vectors, the second is for the operation of vector and matrix, the last layer implements the operation among matrixes. The upper layer is founded on the lower layer.

3.1. HPL Arithmetic Brief Description

The software package is used to solve *N*-dimension linear equations Ax=b. First it should select local column principal component method to decompound coefficient matrix as the following form.

$$[A,b]=[[LU],y] \qquad (1)$$

In the decomposition, the exchange of L factor of lower triangular matrix is applied on b gradually, so the resolution of x can be achieved through solving the linear equations Ux=y which U is the upper triangular matrix. In order to guarantee nicer load balancing and arithmetic expandability, data as circulant block forms is distributed to two-dimension girds which are composed by the number of P×Q processes. The N × (N+1) scale coefficient matrix is divided into many NB×NB scale data blocks, then these blocks are distributed to P×Q scale processes girds in rotation turn. The work goes on both row and column directions.

Here N, NB, P and Q can be modified at any time according to cluster practical configure and user' need. These are very important and critical parameters in HPL test. Beside these, HPL also provides many convertible parameters which can be set during amending HPL.dat before test.

3.2. Factors of Influence Performance

BLAS: BLAS is the core of realization high performance [14] of vector matrix calculation. HPL contains large of float-point operations which are actualized by calling BLAS library.

Internet: The internet of cluster system is very important for the whole system performance because the communication parts always become the bottlenecks. It must use special high-speed interconnection to meet the calculation requirements. The same parameter configures are set on two different communication network to have HPL test, the results show that high-speed network makes the performance enhancement much.

The test parameters: The parameters selected in the HPL test have a lot to do with the result. We can learn every parameter meaning from HPL algorithm. First, the meaning of scale N is very important to user. Here N is the dimension of A in the linear equation Ax = b, and its maximum depends on the available memory. The program computation of HPL test uses matrix of 64 bit precision. So N is supposed to the problem space, M is the quantum of memory (units MB), N and M have the following relations:

$$N = \sqrt{M \times 1024 \times 1024 \times 8 \div 64} \approx 362\sqrt{M} \qquad (2)$$

Supposed the cluster system has totally P computation nodes, the average available memory of every node is m, and the value of problem space can be achieved from the below formula.

$$N = 362\sqrt{m \times 0.8 \times P} \qquad (3)$$

The formula shows that the platform uses approximate 80 percent of total memories [18] to make the Linpack test, and the left is provided to other processes.

3.3. NB-the Size of a Block Matrix

According to HPL algorithm, coefficient matrix is divided into circulants of square of NB. These circulants are allotted to processes. The size of NB is the granular computing which greatly affects the performance of computation. Twenty-four processes are created on six nodes, and HPL tests are made for three different problem sizes with four diverse NB values. The results are as showed in Figure 2. From the figure to see, it is suitable to adopt the number 192 as the size of matrix blocks. That the value of NB is too big or too small would make the performance descend [15]. The optimal value of NB is unfixed, it relates to idiographic test environment and needs many experiments to achieve.

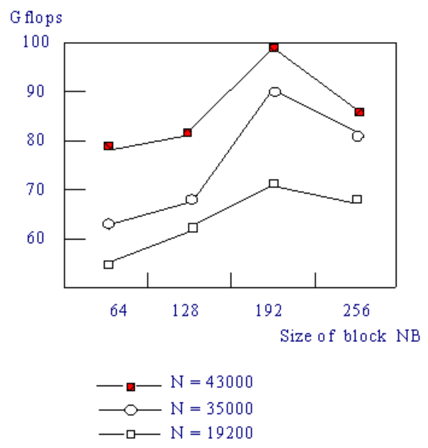

Figure 2. The Influence of Block Size to Performance of Cluster System

3.4. PXQ- distribution of Process Gird

The distribution of process grid would affect the distributing of data on every processor [16]. What kind of process distribution is better depends on the practical communication network architecture. It tests six different distributions on twenty-four processors. The results are showed in Figure 3. From the figure we can see, only the value of P is closed to Q, would the result of test become better. At the same decomposition conditions, the result of P <Q would be better than P>Q.

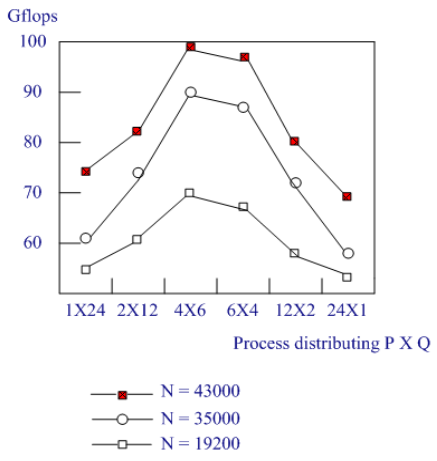

Figure 3. The Influence of Distributed Processes to Performance of Cluster System

3.5. The Correspondences between Process and Processor

When running parallel program, every processor is assigned a process. But for the cluster which is made up of SMP nodes, every node has more than a processor. Given a node owns s processors, there are m nodes, the processes should be m* s. the serial number of process is 0,1,…, s*m-1. So the process and processor can have different corresponding relation.

4. The Summarization of Analysis for Test Results

HPL benchmark test is one of the important testing in all performance tests of cluster system. During the real analysis test for small scale of six nodes, we can draw the following conclusions. One is that we should use approximate eighty percent of the sum of memories to make HPL test. The second is that for the cluster system of given computer nodes , the high performance can be achieved when the dimensions of processor P and Q become closer (P<=Q).

The experiment results show that the factors affected cluster performance include hardware and software. The hardware comprises main frequency of processor, capacity of EMS and bandwidth latency of interconnection and so on. The software includes hit rate of Cache, cost in communication. For a special cluster system, the hardware is confirmed, so the work of optimization should potentially exert hardware performance, and reduce the cost of software at the same time. In terms of another angle, cluster system is made up of many nodes linked by means of relaxed coupling. So we can analyse the performance from single node, and multi-nodes coordination.

4.1. Efficiency of Single Node

To improve the performance of cluster system, we first don't think about communication, and optimize the single node to quicken the calculation speed. speed peak value of processor is determined by main frequency and float-point times in every period. If the speed peak value is affirmatory[17], the optimization aim is to make real speed approach the peak value. Besides processor, physical capacity of EMS have some influence on calculation efficiency. Specially to SMP node, EMS becomes more larger, the conflict of CPU accessing EMS would get more smaller. This would enhance the speed of program. During using two SMP nodes test, it shows that if EMS capacity increases from 512M to 1GB, the calculation efficiency advances from 76.3 to 80.5. In the cluster system, we use approximate eighty percent of the sum of memories to make HPL test.

4.2. Communication among Nodes

Communication is the bottle-neck of cluster system because it is an extra charge except computation. Any change of software would affects the system performance because it impacts on proportion of cost in communication. There are some conditions we should notice. The first is how to select question scale N. Question scale decides the size of computation. N becomes more and more larger, the cost of time would becomes longer and the occupied space would need more. If N exceeds a fixed value, the EMS would be shortage and need disk exchange. It would greatly reduce the calculation efficiency. Before selecting N, we should estimate the value according to formula mentioned before. Under the critical value, the larger N is, the higher performance is. Because N becomes large, the cost of size of computation increases than the communication.

The second is the selection of NB. From the point of view of data distribution, the smaller NB is, the more block are, it is advantageous to load balancing. From calculation angle, if NB too small, it would limit the system performance because the reuse of data in the highest of storage structure is few, and the information required exchange increases. The optimal value of NB has related to N, and also related to performance ratio of calculation/communication . The measurement of NB is optimal or not depends on whether the communication spending reaches the least. From the communication angle, if NB is too small, it would greatly increase the traffic among processes and decrease the proportion of calculation. If NB is too big, it would influence the load balancing of processors and increase the traffic spending. So if calculation performance is superior than traffic, it is better to select the big NB value, contrarily to select small value .

The third is to select the processor grid distributing. From the test results, it is most efficient when P approaches Q and P is less than Q[18]. That P is close to Q is propitious to load balancing. Communication keeps balance in row, column direction. If not, it would have large messages transfer in a pair of processors. Based on this assumption, P should be less than Q because column traffic is large than row according to Linpack arithmetic. If column processes are a bit more, it would decrease the traffic spending.

The fourth is the corresponding relation of processor and process. When selecting the corresponding relation, we should consider the program communication model character. For Linpack, matrix is saved as the form of transposition and communication is much among columns in the decomposition of Panel. The blocks in the same column are assigned to a processor[19], it would cause much process communication in a pair of nodes and incur network ports competition, block and wait. This would waste some times.

5. The Parallel Computation of Openfoam on Cluster System

5.1. Partition Algorithm

The realization of parallel computation of OpenFOAM on cluster system is that decomposes fluid field grid into many subdomains, and the number of which is an exact multiple of computation node (for example, eight subdomains may correspond to one, two, four, eight node). Every subdomain would stay different computation node [20]. It may be the node of parallel machine or the program which run on many CPU platforms. The concurrent principle is described as the following. First it makes fluid field divided into many subdomains , and assigns them to CPU. Then it loads initial flow field information, geometric information of subdomains into memory of CPU, starts up computation processes in every CPU. The main process dispatches CPU to calculate. In the course of every step iterative solution, CPU completes the calculation of submain and data exchange on the border [21]. Before the parallel computation, it needs define the node which will be used. The node definition is completed in the example, and under the example directory which name is machines, it includes node1, node2 and so on. A application program based on parallel computation uses the order, *mpirun*, the command is as following:

```
mpirun --hostfile <machines> -np <nProcs> <foamExec> -parallel > log
```

5.2. Example and Test

It will introduce reactingFoam which is one of OpenFOAM solvers to seek the solution. ReactingFoam [22] is a solver for turbulent combustion model based on PASR[23]. The combustion model used in the numerical simulation process is $k-\varepsilon$ double equation model. The model can not only give satisfactory results for some ordinary flow situations such as flat wall boundary layer, weak plane jet and tube flow, but also need small amount of calculation. The double equation is shown as following.

$$\rho\frac{D\varepsilon}{Dt}=\frac{\partial}{\partial x_i}[(\mu+\frac{\mu_t}{\sigma_k})\frac{\partial\varepsilon}{\partial x_i}]+C_{1\varepsilon}\frac{\varepsilon}{k}(G_k+C_{3\varepsilon}G_b)-C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \quad (4)$$

$$\rho\frac{Dk}{Dt}=\frac{\partial}{\partial x_i}[(\mu+\frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_i}]+G_k+G_b-\rho\varepsilon-Y_M \quad (5)$$

Above the equation, Gk represents the tubulence energy produced by mean velocity gradient and Gb expresses the tubulence energy produced by the effects of buoyancy. YM shows the influence of compressible turbulent expansion speed on the total dissipation rate. The turbulent viscosity coefficient is below:

$$\mu_t=\rho C_\mu\frac{\varepsilon^2}{k} \quad (6)$$

The chemical reaction mechanism:The chemical reaction mechanism plays a decisive role in the chemical kinetics of combustion [24]. The mechanism we used is GRI3.0 database with chemical reaction which is the result of linkage OpenFOAM and Chemkin. It Relates to the 53 groups of gas and combustion air which consist of 325 elementary reactions. The kinetic parameters can be found in the literature [25]. For the fuel CmHn, a general reaction with oxygen as below following:

$$4C_mH_n+(2m+n)O_2\rightarrow 4mCO+2nH_2O \quad (7)$$

For CO, the general reaction with oxygen is:

$$2CO+O_2\leftrightarrow 2CO_2 \quad (8)$$

The numerical calculation method: To solve the multicomponent flow conservation equation describing the combustion process, we adopt the fully implicit finite volume method based on SIMPLE [26]. In the simulation, the diffusion, convection, time difference scheme are used respectively for two order linear modified Gauss difference scheme, two order upwind and the first-order implicit Euler difference scheme. The solving process of SIMPLE algorithm is shown in Figure 4.

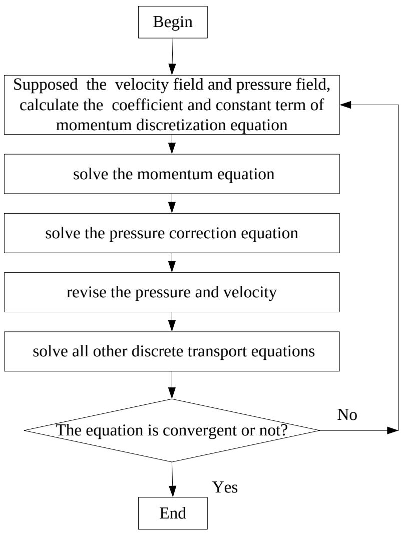
Begin
Supposed the velocity field and pressure field, calculate the coefficient and constant term of momentum discretization equation
solve the momentum equation
solve the pressure correction equation
revise the pressure and velocity
solve all other discrete transport equations
The equation is convergent or not?
No
Yes
End

Figure 4. Schematic Diagram of the SIMPLE Algorithm for Solving Process

The combustion reaction solver: The construction of turbulent combustion reaction solver is based on the SIMPLE implicit finite volume method .

Calculator includes Make directory, the header files and the main program,total in three parts.The main program is reactingFoam.C, and Make directory contains Option files and Files. Files are used to specify the file to be compiled, here does not include the header file. The Option files define the compiler options, for the specified header file location and dynamic library .The header files include hEqn.H, pEqn.H,UEqn.H and YEqn.H [27].

5.3. The Simulation Results and Analysis

The initial and boundary conditions: On the platform of OpenFOAM, we take the numerical simulation for the gas and air turbulent diffusion combustion reaction in the specific combustion chamber under grid environment. The solution to calculation of regional is axisymmetric problem. It is part of a bluff body lied from the bluff body axial division 5 ° area. The Blunt radius are 50 mm, and the fuel entrance radius is 36mm.The structure of bluff body adopts structured grid discrete computational domain, which contains 5580 grids. ReactingFoam is a grid structure which is 0.1 meter long ,0.1 meter wide and 0.1 meter high. It is filled with air and on the top of the model, there is a syringe which can inject CH4 into combustion chamber. In ReactingFoam, liquid drop is evaporating and burning. The Fig.5 is the geometry shape of reactingFoamcounterFlowFlame2D. The initial conditions of the reaction is shown in the following table 1.

Table 1.Combustion Boundary Conditions and Initial Conditions

BC	FuelInlet	AirInlet	InternalField
T/K	900	900	300
$V(m \bullet s^{-1})$	118	40	—
$p/10^{\sqrt{5}} P_a$	30	30	30
CH_4	0.13	0	—
O_2	0	0.21	0.21
N_2	0.02	0.78	0.78

CO	0.34	0	—
H_2	0.05	0	—
Ar	0	0.01	0.01
CO^2	0.25	0	—
H_2O	0.21	0	—

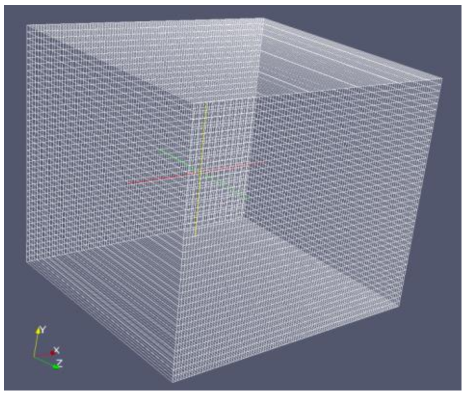

Figure 5. Geometry Shape Diagram

5.4. The Computational Process and Results of Analysis Calculation

At the stage of numerical simulation, we make the reactants and accompanying flow from fuelInlet and airInlet respectively pour into the combustion chamber at the speed of 118 m/s and 40 m/s. During 216000 seconds computation, the combustion reaction is stable. The instruction has been introduced in partition algorithm. Figure 6 and Figure 7 show that CH4 reaction process, the final burning state and the temperature of combustion chamber when CH4 burned up. We can get the basic simulation result of combustion chamber, and the imitative effect is very good. Distribution of cloud better reflects some basic characters of reacting flow field. The experimental temperature distribution is identical with Sydney obtuse HM1 [28] experiment result of flame, especially good results corresponding to the highest temperature obtained from the combustion effect.The recirculation zone formed by the fuel jet diffusion does not exit in the flame upstream,which makes the simulation results bias to the laminar burning. Therefore, the set-up of k and ε in the $k-\varepsilon$ double equation model need further study so as to get a better simulation effect.

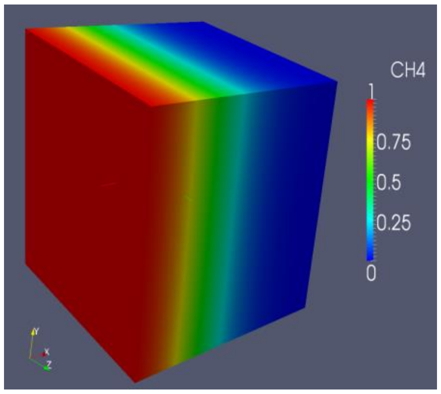

Figure 6. CH4 Mass Fraction Distribution Cloud Imagery (H2O/mol)

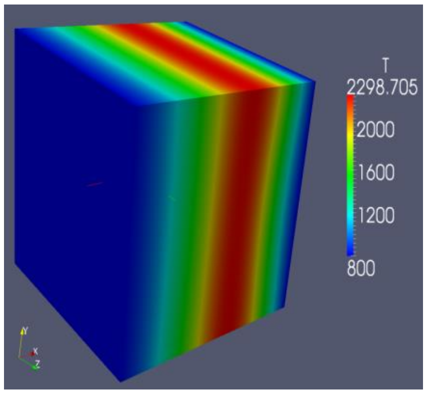

Figure 7. Temperature Distribution of Combustion Reaction (T/K)

5.5. The Analysis of Parallel Effect

Before computation, we should separate the grid of combustion into N subdomains assigned to CPU. Then the initial related information of subdomains is loaded on corresponding CPU memory [29]. Every kernel of CPU separately starts up a computation process. So there are N processes having parallel calculation at the same time. Figure 8 shows the relation of time consumed on computation and the number of nodes. From the figure, we can get the information that as the nodes increase, the time of computation becomes quick. This illuminates that the OpenFOAM platform represents better after it coupled high performance parallel computation [30]. It should be further researched.

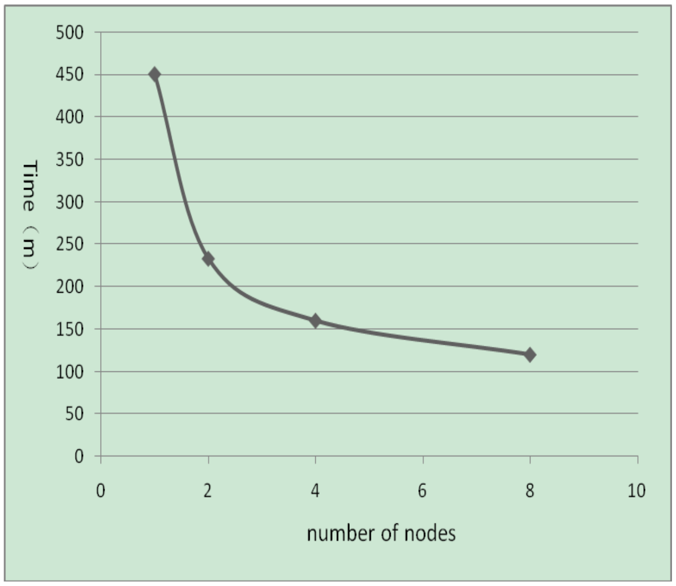

Figure 8. The Acceleration Rendering of Parallel Computing Platform

6. Conclusions

The paper conducts various surveys about important factors which affects greatly the cluster system performance. It designs a solver based on the turbulent combustion flow model and has successfully construct a numerical simulation on parallel computation and OpenFOAM platform. It also structures a solver based on PaSR model. During the numerical simulation about combustion flow, it realized hydromechanics parallel numerical simulation by using many computation nodes. The results show that the cluster system has excellent performance, stability and efficiency. It provides efficient, time-saving services. OpenFOAM platform coupled high performance parallel computation represents sound.

Acknowledgements

This work was financially supported by the natural science research project of the science department of Henan province (122300410397,132300410276).

References

[1] P. Wang, S. Lǖ and Z. Nie, "Parallel Computation Application and Practice", China Machine Press (**2008**).
[2] J. F. Wendt, "Springer: Computational Fluid Dynamics", AN INTRODUCTION (**2008**).
[3] S. Shepler, B. Callagham, D. Robinson, and R. Thurlow, "NFS version 4 protocol", RFC3010, Network Working Group, December (**2000**).
[4] T. Ylönen, "Proc. of the *Sixth USENIX Security Symposium*", vol. 12, (**1996**), p. 37.
[5] E. Gabriel, "OpenMPI: Open Source High Performance Computing", http://www.open-mpi. org/(2011).
[6] H. Jasak and M. Beaudoin, "OpenFOAM Tubro Tools: From General Purpose CFD to Turbomachinery Simulations", Proceedings of ASME-JSME-KSME Joint Fluids Engineering Conference (AJK2011-FED) (**2011**).
[7] X. Wang and Z. Du, "Computer Science", vol. 32, (**2005**), p. 11.
[8] OpenFOAM Company, OpenFOAM User Guide, Berkshire: OpenCFD Company (**2010**).
[9] N. E. L. Nielsen, "Numerical investigation of a BFR using OpenFOAM", Fluids and Combustion Engineering (**2008**).
[10] R. J. Kee, F. M. Rupley and J. A. Miller, "CHEMKIN Release 4.1.1.Reaction Design", San Diego, CA (**2007**).
[11] G. Coulouris, J. Dollimore, T. Kindberg and G. Blair, "Distributed Systems: Concepts and Design", 5th ed, Addison Wesley Press, (**2011**), pp. 1-63.
[12] F. A. Cummins, "Enterprise Application an Architecture for Enterprise Application and Systems Integration", New York: John Wiley & Sons, Inc., (**2002**), pp. 2-35.
[13] X. C. Lu, H. M. Wang, J. Wang, J. Xu and D. S. Li, "Internet-Based virtual computing environment: Beyond the data center as a computer", In Proc. of the Future Generation Computer Systems, vol. 1, no. 29, (**2011**), pp. 309-322. [doi: 10.1016/j.future.2011.08.005]
[14] D. E. Bakken, "Middleware, Encyclopedia of Distributed Computing", Kluwer Academic Press, (**2003**). http://www.eecs.wsu.edu/2000~bakken.
[15] Object Management Group, Common object request broker architecture: Core specification v3.0.2. (**2002**).
[16] M. P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-Oriented computing: A research roadmap", In Journal of Cooperative Information Systems, vol. 17, no. 2, (**2008**), pp. 223-255.
[17] Y. V. Natis, "Service-Oriented Architectures Scenario", Gartner, (**2003**), pp. 1-6.
[18] K. Geihs, "Middleware challenges ahead", Computer, vol. 34, no. 6, (**2001**), pp. 24-31.
[19] P. A. Bernstein, "Middleware: A model for distributed system services", Communications of the ACM, vol. 39, (**1996**), pp. 86-98.
[20] H. Espanha, J. P. Nagh, P. Shtabrizi and Y. Xu, "A Survey of Recent Results in Networked Control Systems", Proceedings of the IEEE, vol. 95, no. 1, (**2007**), pp. 138-162.
[21] A. Iyengar, "WebSphere Business Integration Primer: Process Server, BPEL, SCA, and SOA", IBM Press, (**2007**), pp. 1-67.
[22] T. Barnes, A. Messinger, P. Parkinson, A. Ganesh, G. Shegalov, S. Narayan and S. Kareenhall, "Logging last resource optimization for distributed transactions in Oracle WebLogic server", In: Proc. of the 13th Int'l Conf. on Extending Database Technology. New York, (**2010**), pp. 651-656.
[23] E. Meijer and G. Bierman, "A co-relational model of data for large shared data banks", Communications of the ACM, vol. 54, (**2011**), pp. 49-58.
[24] H. Y. Zhou, W. Wang and W. B. Zhang, "Self-Adaptive adjusting approach for cluster in cloud computing", Journal of Frontiers of Computer Science and Technology,(**2011**), vol. 5, no. 4, pp. 347-355.
[25] C. A. Waldspurger, "Memory resource management in VMware ESX server", ACM SIGOPS Operating Systems Review, vol. 36, (**2002**), pp. 181-194.
[26] P. Barham, B. Dragovic, K. Fraser, "Hand Xen and the art of virtualization", In: Proc. of the 19th ACM Symp. On Operating Systems Principles (SOSP 2003), New York: ACM Press, (**2003**), pp. 164-177.

[27] P. Lama and X. Zhou, "Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee", In Proc. of the 18th Annual IEEE/ACM Interl Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEEComputer Society, (**2010**), pp. 151-166.
[28] N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman and R. Wolski, "AppScale Design and Implementation", (**2009**).
[29] A. Bhuvan, S. Prashant and R. Timothy, "Resource overbooking and application profiling in shared Internet hosting platform", USAACM Trans. on Internet Technologies, vol. 9, no. 1, (**2002**), pp. 239-254.
[30] X. Liu, J. Heo, L. Sha and X. Y. Zhu, "Queuing-Model-Based adaptive control of multi-tiered Web applications", IEEE Trans. on Network and Service Management, vol. 5, (**2008**), pp. 157-167.

Authors

Yiran Wang, he was born in 1976 at He Nan province, China, Yiran graduated from ZhengZhou University in 2005 and attained the degree of master of computer science. His major field is computer application, network and enterprise information. He now works in ZhouKou Normal University. He teaches many core courses of computer science such as data structure, C language, principle of micro-computer and so on. Now, the major research is Internet of things. Mr. Wang is now an associate professor in school of computer science and technology of ZhouKou Normal University.

Zhou Wen-Gang, he received B. Eng Degree in computational mathematics from HeNan Normal University and the M. Eng Degree in computer application technology from North China University of technology in 1997 and 2007 respectively, He is currently researching on the analysis and design of Intelligent algorithm