P2P Computing Mode and Its Application in Flow Computation

Fang Hualiang[1], Liu Junhua[2], Chen Zhongwei[3]

(1College of Electric Engineering, Wuhan University, Wuhan, P.R.China. 2 Shangluo Power Bureau, Shanxi, P.R.China. 3 Huarong Power Bureau, Hunan, P.R.China)

ABSTRACT: The P2P computing mode which may effectively overcome some shortages of the present sequential and parallel computing mode is applied to the large-scale power flow computation in this paper. The advantages of the P2P mode are analyzed and presented. The mathematic model and computing method is put forward and analyzed based on current-influx model. Based on current-influx method, power network is easily partitioned, and the relation among subtasks becomes weak. The mutual dispatch management in P2P are researched for the flow computation. The static and dynamic dispatch methods are adopted, which may flexibly adjust the topological structure of computing network. The dispatch arithmetic of the load-balance is studied, which may transfer high load to near computing node according to the total present computing instance. And the computing efficiency is enhanced totally. The proposed computing model and methods are implemented in IEEE 118-buses system. The computing results have approved the validity of this computing model.

KEY WORDS: P2P computing, Network computing, Power flow, Dispatch arithmetic

I Introduction

The flow computing is considered the fundamental element of modern power system control centers. It can be defined as a calculation program, by which the operation state can be solved. The sequential computing may meet current requirements of flow computation, by improving and developing in the past few decades. However, there are some difficulties to deal with on-line, real-time, optimal flow in large-scale power system[1-2].

The P2P computing technology is introduced and studied in power applications. Unlike dedicated multiprocessors or high performance computing platforms, the proposed method is based on an existing Local Area Network. Apart from exceptional circumstances when all the computers are fully utilized, most of them are either idling or not loaded to their full capacities. In P2P, peers act as both clients and servers, form an application-level network, and route messages[3].

This paper studied the computing method and dispatching algorithms by means of P2P in power flow computation. The organization of the paper is as follows. The advantage of the P2P is presented in section 2. The mathematical model and distributed arithmetic of flow are given in section 3. The dispatch algorithm is studied in section 4. The simulation results of flow are shown in section 5. Conclusions are presented in section 6.

II The Advantage of P2P

At present, the underlying impetus for research is to improve the performance of power system applications to provide real-time control and support for proactive decision making in an economical and secure way. Huge computing capability pooled in P2P platform, will solve the above problems. The distributed flow may be realized by using P2P technology, which can integrate existing computing capability. The computing result can store in local dispatcher centre, which is transmitted and shown between dispatcher centre quickly[4-7].

The advantages of the P2P mode meet the computation need. The characteristic of electric network is the wide-area distributed, hiberarchy management, etc. These are very suitable for P2P computing mode. Compared with several parallel computation modes, it has some special advantages. Parallel computation can improve the speed, by assembling all data of the whole region. However, these data have certain business secret. In P2P mode, the calculation program and outside equivalence parameters are submitted to the region with data, only results are exchanged among the regions. So the data sharing, better commercial secret and little data communication are realized easily. The scale of parallel computation can't be expanded limitlessly, and can not catch up with the demand of computing question. The P2P mode may obtain limitless computing capability theoretically and execute real-time even over real-time simulation computation. So it will solve the real-time problem of huge-scale power system effectively[8-9].

III The Mathematical Model and Distributed Arithmetic of Flow

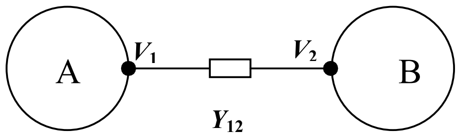
In general, the flow computing may be formulated as follows:

$$Y \cdot V = I = V_D^{-1^*} \cdot S^* \tag{1}$$

where Y is admittance matrix of network; V is node voltage vector, $V_D = diag\{V_1, V_2, \cdots, V_n\}$; I is node current vector.

The current-influx method is adopted, which is solved by iterative arithmetic. The last voltage $\boldsymbol{V}^{(\mathbf{k-1})}$ replace the right $\boldsymbol{V}$, and $\boldsymbol{V}^{(\mathbf{k})}$ replace the left $\boldsymbol{V}$. So the linearization equation group is formulated as follows:

$$Y \cdot V^{(k)} = I = (V_D{}^{(k-1)^*})^{-1} \cdot S^* \tag{2}$$



Figuer 1 Interconnected power system

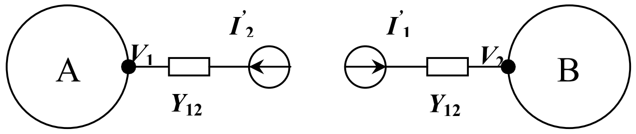


Figure 2 Decomposition of interconnected system

The tasks need to be partitioned by the thick grain size in the P2P computing. The partition method based branch was adopted in this paper. By this method, the size of each subarea is almost equal. The communication among computing is reduced with few cutset branches. The network branch is cut in Fig.1, as shown in Fig.2. It is very complicated to process the border branch accurately. Here, the border voltage can be expressed with the neighboring voltage (replaced with last iterative value). The border current can be shown as follows:

$$I_{21} = (V_2 - V_1)Y_{12} \tag{3}$$

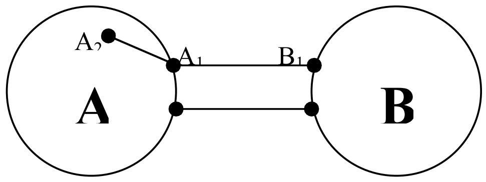


Figure 3 Rectifying chart of sub-region

It is very easy to divide computing task statically based on the current-influx. P2P may carry on peer dispatch and manage. The network regions are adjusted, and get more optimal regions statically, fast convergence and high computing efficiency. Based on the current-influx, the border voltage were exchanged among subregions, the convergence of which effect the global convergence. After several iterative steps, some voltage variables with worse convergence need to be processed. In Fig.3, the border node A_1 with worse convergence of voltage would be distributed to the subregion A, and the inner node A_2 with better convergence would become border node. The border branch A_1B_1 with worse convergence of current would be distributed to the inside of subregion A, and the branch A_2A_1 with better convergence would become the border branch. The procedure was shown in Fig.4.

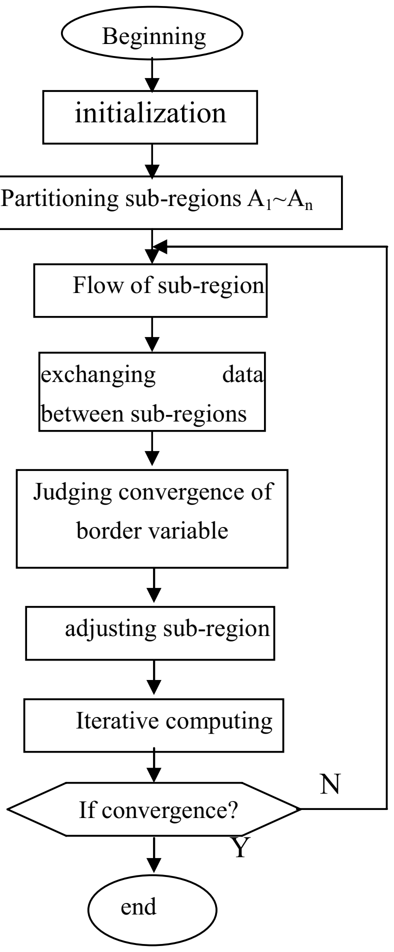

Beginning
initialization
Partitioning sub-regions A_1~A_n
Flow of sub-region
exchanging data between sub-regions
Judging convergence of border variable
adjusting sub-region
Iterative computing
If convergence?
N
Y
end

Figure 4 Flow chart of flow computation

IV The Dispatch Algorithm in P2P

Because the environment of network computing is very unstable, the relation between the task and computing resource need to be coordinated. The good dispatcher algorithm might utilize computing resources effectively and improve computational efficiency.

Generally, the initial partition of tasks was not optimal, and were adjusted dynamically. The calculation amount of each sub-task (t_i) could be estimated by the network node number, and the communication could be estimated by the branches between the subregions. These might be described with DAG (directed acyclic graph). The static node information of DAG was described with *ST*(*ID, T_{init}, CompRequire, D_{eg}, ID_{pre}, ID_{next}*). ***ID*** was the mark of task node. ***T_{init}*** was the initial estimation time of the task. ***CompRequire*** was the computing requiring. ***D_{eg}*** was the node degree. ***ID_{pre}*** and ***ID_{next}*** were father node and son node respectively. The dynamic dispatch information might be described as ***DDI***(*RID, Iter, Finished, Rload, CanReceive*). where ***RID*** was resource ID; ***Iter*** was the number of iterative times; ***Finished*** was the finished sign of present task; ***Rload*** was the load of computing resource; ***CanReceive*** expresses whether other tasks could be received.

1.Static Dispatch

After the task is submitted, P2P platform analyzed task. The tasks were assigned initialize according to the resource information table. Assume the number of sub task is N_t, and the number of computing node is N_c. There are the following several situations:

(1) When $N_t > N_c$, some sub-tasks need to incorporated. Assume $N_{ave}=[N_t / N_c]+1$. Some adjoining nodes combined into a local connected graph according to ***ID_{pre}*** and ***ID_{next}*** of sub-task in DAG, and a new task node came into being. The size of new node and ***ST*** were obtained according to ***N_{ave}***.

(2) When $N_t = N_c$, the number of sub-task is equal to that of computing node. Some sub-tasks with small ***T_{init}*** were incorporated into tasks with big size, according to **max(*T_{init}*)**. Some computing resources were left, to carry on flexible dispatch dynamically.

(3) When $N_t < N_c$,, there is sufficient computing capability. Every sub-task could obtain more one computing resource. Some important sub-tasks could realize redundant calculation, and the better dependability and fault-tolerant were achieved.

2.Dynamic Dispatch

The inside and outer exchanges were carried on between tasks after one iterative computing mutually. The last actual time was expressed with ***T_{finish}***. The next execution time was expected:

$$T^{(k+1)}_{\exp ert} = \sigma \cdot T^{(k)}_{finish} + (1-\sigma)T^{(k)}_{\exp ert} \tag{4}$$

where σ is a change coefficient, and change with the node load rate.

In each iterative, the accomplishment ratio of task is:

$$\lambda_{\exp} = \frac{T^{(k)}_{finish}}{T^{(k)}_{\exp ert}} \tag{5}$$

The average of the last k times is:

$$\overline{\lambda}_{\exp}(t_i) = \sum_{i=1}^{k} \lambda_{\exp}(t_i) \tag{6}$$

The variable with worse convergence was adjusted, new ***T_{init}*** of every sub-task was obtained.

In iterative procedure of flow computation, one node with finished task sent ***finished*** signal to adjoint computing node at once. At the same time, its own ***CanReceive*** was locked. CanReceive in the adjoint node is reduced 1 after receiving ***finished***, and wrote down the other ID and Rload. After arranging in an order from small to large according to ***CanReceive*** value, the load with minimum ***CanReceive*** was the heaviest. Its partial task was distributed to the computing node, and the load is more balanced among sub-tasks.

The average finished time of one iterative computing is:

$$\overline{T}_{finish} = \sum_{i=1}^{n} T_{finish}(t_i) \tag{7}$$

The average finished time of the next iterative computing is estimated:

$$\overline{T}_{\exp ert} = \frac{1}{n}\sum_{k=1}^{n} \overline{\lambda}_{\exp}(t_i)\overline{T}_{\exp ert}(t_i) \tag{8}$$

The computing amount in the node with finished task might be estimated:

$$L_{add}(t_i) = \frac{\overline{T}_{\exp ert} - T_{finish}(t_i)}{T_{finish}(t_i)} L_{init}(t_i) \tag{9}$$

The computing amount ($t_{low}(j)$)in the node with heave load might be transmitted:

$$L_{sub}(t_{low}(j)) = \frac{T_{finish}(t_{low}(j)) - \overline{T}_{\exp ert}}{\overline{T}_{\exp ert}} L_{init}(t_{low}(j)) \tag{10}$$

The node, the computing time of which was less than $\overline{T}_{finish}$, was executed according to (9), otherwise (10). So the task was redistributed, the balance of load was better in next iterative.

V The Simulation Computation Results

IEEE118 bus test system run on the P2P platform. IEEE118 bus system is divided into four sub-regions A_1、A_2、A_3、A_4。

A_1＝{1-17, 25-32, 113-115, 117}

A_2＝{18-23, 33-44}

A_3＝{24, 45-79, 116, 118}

A_4＝{80 -112}

The branches between A_1 and A_2＝{(15, 33), (15, 19), (17, 18), (32, 23) , (25, 23)}

The branches between A_2 and A_3＝{(23, 24), (38, 65), (44, 45), (42, 49) }

The branches between A_3 and A_4＝{(77, 82), (97, 80), (81, 79)}

At first, the sequential computation was executed in one machine. The result was that t =3.2s, the iterative number was 35.

At the second experiment, in simulation P2P network environment, the balance strategy of computing load was used by adjusting the subregions. So more optimal dispatch was obtained. The computing result was that ***t*=2.5s**, and the iterative number was 32.

Because of the complexity and unstability of the network environment, the result of the total computing time is not very obvious. In addition, the number of computing tasks was little. The P2P mode will have more obvious effet while the number of the tasks is very big. In the iterative computing, the subregion was adjusted according to the convergence of the border variable, and better convergence was obtained. So the computing performance was improved.

VI Conclusion

The P2P This computing mode may obtain computing resources of low cost easily, and has better cross-platform, community, and scalability. It can increase computing power through integrating limitless resources in network. These meet the computing demands of huge-scale power system. In this computing mode, peer dispatcher, static and dynamic dispatcher are carried on, which make it more easy to balance of load. In the iterative procedure of flow computing, computing resources are dispatched dynamically. So the topological structure of resource may be changed flexibly, in order to meet the needs of the task. The P2P computing mode improved the computing speed and efficiency of the flow in this paper. So it will become one of the most effective methods to solve the simulation computation in huge-scale power system.

VII References

[1] YU Jilai, WANG Jiang, LIU Zhuo. Improvements on usual load flow algorithms of power system[J]. Proceedings of the CSEE. 2001, 21(9): 88-93.

[2] TANG Yong. Present situation and development of power system simulation technologies[J]. Automation of electric power systems, 2002, 26(17): 66-70.

[3] Talia. D, P. runflo. Toward a synergy between P2P and grids[J]. Internet Computing, 2003, 4(7): 94 – 95.

[4] Damiani. E, De Capitani, Di Vimercati, S.etc. Managing and sharing servants reputations in P2P systems[J]. Knowledge and Data Engineering, IEEE Transactions, 2003 ,4(15):840 – 854.

[5] Schmidt. C, M. Parashar. Enabling flexible queries with guarantees in P2P systems[J]. Internet Computing, 2004, 3(8):19 – 26.

[6] Mischke. J, B. Stiller. A methodology for the design of distributed search in P2P middleware[J]. Network, IEEE, 2004, 1(18):30 – 37.

[7] Lienhart. R, Holliman. M, Yen-Kuang Chen etc. Improving media services on P2P networks[J]. Internet Computing, 2002, 1(6):73 – 77.

[8] GU Wei , HUANG He , JIANG Ping. Research on the Model of UPFC for Three Phase Power Flow Calculation[J]. High Voltage Engineering, 2005, 31(3):71-73.

[9] HUANG Yanquan, XIAO Jian, LIU Lan, JIANG Gonglian, HAN Huarong. A coordinational parallel algorithm for power flow calculation based on branch cutting. Power System Technology, 2006, 30(4):21-25.

VIII Biographies

Fang Hualiang received his Ph.D in Department of Electrical Engineer, from Huazhong University of Science and Technology, in 2006. He is working in Wuhan University. His research interests are application of parallel and Grid computation to power system.
E-mail: hl_fang@21cn.com.
Address: College of Electrical Engineering, Wuhan University, Wuhan, Hubei, P.R. China, 430074.

Liu Junhua received his received his B.Sc. degrees in Department of Electrical Engineer, from Huazhong University of Science and Technology, in 1999. He is working in Shangluo Power Bureau. His research interests are application of information technology to power system.
E-mail: ljh7588@sina.com.
Address: Shangluo Power Bureau, Shanxi, P.R. China, 726000.

Chen Zhongwei received his received his B.Sc. degrees in Department of Electrical Engineer, from Huazhong University of Science and Technology, in 1999. He is working in Huarong Power Bureau. His research interests are application of information technology to power system.
E-mail: flylpp@yahoo.com.cn.
Address: Huarong Power Bureau, Hunan, P.R. China, 414200