Two-Phase Query Optimization in Mobile Ad Hoc Wireless Networks

Han Ke
Department of Computer and Information Engineering; Harbin University of Commerce
Harbin, China
hanke@hrbcu.edu.cn

***Abstract*—This paper investigates multi join queries in Ad Hoc wireless networks. A two-phase query optimization method is presented, which optimizes a query with two phases. First, it optimizes middle result of a query and produces optimal join sequence, which makes account of transmission data minimum. Second, the join tree from first phase is separated into several sub queries and distributed to the corresponding data nodes. The data nodes dynamically select nodes to execute these sub queries. Also, this paper presents optimization algorithm of middle result, query separation algorithm, query execution nodes selection algorithm and query plan execution algorithm. The experiment results show that the algorithms not only greatly decrease energy cost of a query but also adapt to characteristics of Ad Hoc wireless networks, including multi hop communication and dynamically changing topology.**

Keywords-MANET; Mobile database; Multi join query; Query optimization

I. INTRODUCTION

Recently, mobile database technology is widely used to application fields, including mobile office, weather forecast, stock activities and mobile conduction in future digital battlefield. The emergence of mobile Ad Hoc network(MANET) brings new type of organization for mobile database. MANET is self-organized, low bandwidth, multi-hop route, energy limitation, and disconnection frequently. Traditional mobile query processing techniques could not apply to mobile database based on MANET, thus we should study the new query processing technique.

In 1998, Hans-Erich Kottkamp proposed location-aware query processing method and join algorithm between two relations based on location information[1], which did not refer to the energy and bandwidth of mobile nodes. In 2001, Chang-Hung Lee used remote join to process mobile distributed query, which saved the total data communication cost of query[2]. However, the algorithm supported client and server situation with base station, and it could not apply to MANET. Hüseyin Gökmen Gök studied the location-aware continuous queries[3]. In 2005, Yongluan Zhou proposed a self-adaptive query processing model based on distributed environment[4]. It dynamically optimized a query plan when executing the plan. Each node has a Eddy to listen the network transmission speeds and workloads of processing nodes. It could change the query plan in real time according to the feedback information from Eddy. When optimizing again, it needs large communication cost. So it does not apply to MANET with low bandwidth and limited energy. Jinbao Li presented join algorithm of two nodes based on MANET[5], which could adapt to the characteristics of MANET, such as availability, connectivity, low-bandwidth and data quality. Also it decreases the communication cost between the data nodes and improves the execution efficiency of join operation between two nodes. However, it needs further research for the join algorithm of multi nodes and query optimization.

In the mobile database system based on MANET, the communication cost between nodes is further more than the computation cost. Thus it is important to save the communication cost between nodes. To estimate cost of operation is more difficult during query optimization because of the frequent movement of nodes. The inaccuracy estimation of cost will directly have effect on the query optimization, and even get a query plan with high cost. In addition, the wireless communication between nodes brings more disconnection of nodes and faulty data transmission. So first production of query plan and execution of it does not apply to mobile database management system(MDBMS) based on MANET.

II. QUERY PROCESSING OF MDBMS

An Ad Hoc network is a special kind of wireless network consisting of a collection of mobile hosts(MH). There is no fixed infrastructure in the network. The hosts are connected wirelessly. Each one has mobility, so the topology of network changes continuously. The mobile database system in an Ad Hoc network is a dynamic distributed database system, which is composed of some MHs. Each MH has a local database system. Each node can propose a query as a client and can also process queries from other nodes as a server.

The movement of MH causes that estimation of the execution cost of operator is difficult during the query optimization. For instance, when nodes A and B execute the join operator with two relations, it needs to estimate the communication cost of join. As we know, the communication cost is directly proportional to distance between two nodes. Movement of nodes changes continuously distance of two nodes and makes it difficult to estimate communication cost. This paper proposes two phases optimization algorithm aiming at characteristics of mobile database system, such as quick movement of nodes, wireless connection, the battery power and frequent disconnection. First, we confirm the sequence of join with dynamic programming to optimize the size of middle results of join in order to decrease the energy used in transmission of middle results. Second, we separate the query into some sub queries and distribute them to mobile data nodes.

General Program about science and technology research of Heilongjiang province educational commission with No.11541263 and No.11541074.

978-1-4244-4738-1/09/$25.00 ©2009 IEEE

When the nodes receive sub queries, they will dynamically choose some nodes to execute sub queries and choose the join algorithm according to their location, movement speed and energy to decrease the number of tuples and their transmission distance. In this section, we first introduce the representation model and cost model of MDBMS query plan. Second the two phases query optimization algorithm is described. At last we introduce the execution algorithm of query plan.

A. Representation Model of Query Plan based on Multi-mark Tree

MDBMS represents a query plan with multi-mark tree, which shows not only local energy cost but also communication cost of query. Here, local energy cost includes cost of I/O and CPU. In multi-mark tree, the nonleaf nodes(except the root node) represents relational operator, and the leaf nodes represents the relations in the query. Each node has its own (MID,P). If a node is leaf node, MID is a mobile node identification (ID) where the relation locates, and P is the location of the mobile node. If the node is nonleaf node, MID is a mobile node ID executing the operator, and P is the location of the mobile node. There are two types of mark at the edges. $E(op_i,M_j)$ represents the energy consumed by an operator op_i processing the middle result M_j. The subscript j of M_j represents the node ID where the middle result locates. The root node is the query node. E_result at the edge between the query node and operator represents the energy with transmission of the result to the query node.

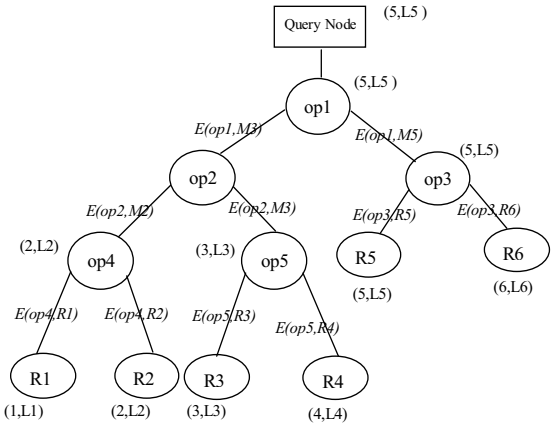

Fig.1 Query plan model in MDBMS

As illustrated in fig. 1, R1 is a leaf node of marked tree, which indicates R1 is a relation of the query. In (1,L1), 1 represents R1 is located in a mobile node identified 1, and L1 represents the current location of the node with description of (X,Y). op4 is the parent node of R1 and R2, which means op4 would be executed on R1 and R2. In (2,L2), 2 represents relation R2 is located in the node identified 2, and L2 represents the current location of the node identified 2. E(op4,R1) and E(op4,R2) at the edges from op4 to R1 and R2 represents the energy cost to transmit tuples of R1 and R2 when executing op4. The operating sequence of the query plan in figure 1 is (R1 op4 R2) op2 (R3 op5 R4) op1 (R5 op3 R6).

B. Cost Model of MDBMS

There are two parts in the cost module of query plan based on multi-mask tree. The first one is the query cost in local data node, including the cost of CPU and disk I/O. The second one is the communication cost among nodes. Suppose query Q contains m operators, the energy cost of executing Q is (1):

$$COST(Q)=\sum_{i=1}^{m}\cos t(op_i)+E\_result \quad (1)$$

Here, $COST(Q)$ represents the total energy cost of query Q, $\cos t(op_i)$represents the energy cost of executing operator op_i, and E_result represents the energy cost of transmitting the query result to the query node. In order to compute $\cos t(op_i)$, we should define a formula (2) as follows :

$$E(op,R)=Byte(Transmit\_Tuple)*Per\_Byte\_Energy*dis(Loc_R,Loc_{op}) \quad (2)$$

Here, E(op,R) represents the energy cost to transmit tuples of relation R when executing operator op. Byte(Transmit_Tuple) represents the total bytes of transmitted tuples. Per_Byte_Energy represents the energy to transmit a byte. Loc_R represents the location of mobile node where R locates. Loc_{op} represents the location of node where op locates. dis(Loc_R,Loc_{op}) is the distance of transmission of tuples. $\cos t(op_i)$ is defined as follows (3):

$$\cos t(op_i)=\sum_j E(op_i,R_j)+cpu\_\cos t+I/O\_\cos t \quad (3)$$

Here, op_i in $\sum_j E(op_i,R_j)$ might be unary operation or multi operation. For instance, the join operator is dual operation, which refers to two relations. Now j equals 2 and it needs to compute the summary of energy to transmit two relations.

The energy cost of CPU and I/O is further less than the one of transmission[6], so we ignore the cost of CPU and I/O. Thus the total cost of query Q $COST(Q)$ is described as follows (4):

$$COST(Q)=\sum_i\sum_j E(op_i,R_j)+E\_result \quad (4)$$

C. Two-phase Query Optimization Algorithm

The two-phase query optimization algorithm is the key component of query processing of MDBMS. The query optimization includes two phases. In the first phase, we aim at the middle result of the query and produce an optimal join sequence described with join tree, which makes the transmitted data minimum. In the second phase, the join tree from the first phase is separated into multi sub queries, and each sub query is assigned the corresponding data node. According to heuristic rules, every sub query chooses the join algorithm and the data node to execute the sub query. Thus the local sub query plan is formed.

1) Optimization Algorithm of Middle Result

When the query node receives a query from the user, the join diagram will be formed after lexical analysis and

syntactical analysis. Next the query node broadcasts the query information. When the data node including the relation, it executes the group operation on the join attribute of the relation and gets the histogram of result to return statistic information to the query node. When the query node receives the statistic information from all the relations, it estimates the size of middle results in order to the accuracy of estimation. At last, we get the optimal join sequence and transform it into a join tree. Here, the separation algorithm of join diagram is as follows:

Input: Join graph G
Output: Information of separation positions
Function: enumerate all possible join sequences, find separation positions with minimal middle result and record in matrix S[I,J]

1. N = LENGTH[vertice(G)] – 1
2. FOR I = 1 TO N DO
   M[I,I] = 0
3. FOR L = 2 TO N DO
   BEGIN
   FOR I = 1 TO N-L + 1 DO
   BEGIN
   3.1 J = I + L -1
   3.2 M[I,J] = ∞
   3.3 FOR K = I TO J-1 DO
   BEGIN
   Compute the size of middle result after joining M[I,K] and M[K+1,J], that is COST(K)
   Q=M[I,K] + M[K+1,J] ＋COST(K)
   IF Q < M[I,J]
   THEN
   M[I,J]=Q
   Record the optimal separation position: S[I,J]=K
   ENDIF
   END
   END
   END

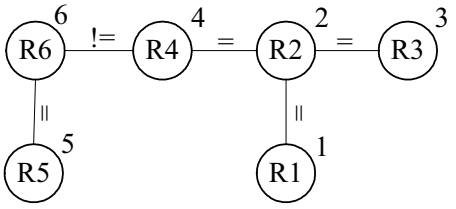

Fig.2 Join graph demonstration

For instance, Figure 2 shows a join graph example of MDBMS. They are equal joins between R1 and R2, R2 and R3, R2 and R4, R6 and R6. It is nonequal join between R6 and R4. Figure 3 is the join tree transformed from figure 2. The left-top sign of a relation is the node ID where the relation locates.

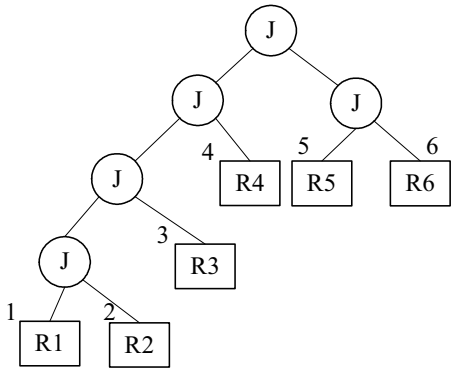

Fig.3 Join tree demonstration

2) Query Separation Algorithm

Before executing the second phase, it firstly calls the query separation algorithm to separate a join tree into several sub queries. During separating, it first preorder travels the join tree from the first phase, and separate the current query from top to bottom until getting to leaf nodes. Figure 4 indicates the result of separating the join tree showed in figure 3. Here sub query 1 depends on the result of sub query 2. Sub query 2 depends on the sub query 3. Sub query 4 is independent and has nothing to do with other three sub queries.

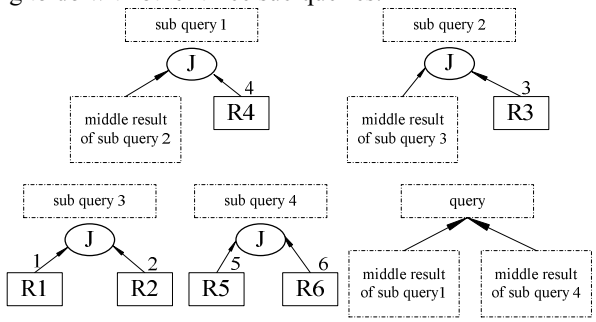

Fig.4 Separation of sub queries demonstration

Separating the query into multi sub queries could completely develop the parallel processing among the sub queries. The sub queries without dependency relationship could be parallel executed to decrease the response time of the query. At the same time, it could balance the load between the query node and the data nodes to reduce the fault possibility caused by the communication energy cost of query node over-consumed. For instance, sub query 3 and sub query 4 in figure 4 could be executed parallely. In addition, each sub query could dynamically optimize according to the current network topology, the location and the moving speed of mobile node to minimize energy cost of local sub query and decrease energy cost of whole query.

3) Query Execution Nodes Selection Algorithm

The query execution nodes selection algorithm mainly optimizes the transmission distance of middle results and reduces the query execution time. Because of moving speed of mobile node changing quickly and variation of neighbor nodes and the topology changing continuously, central optimization of query node would make the estimation inaccuracy and large communication cost. It makes some nodes overload and consume energy rapidly. Also, the query plan after central

optimizing is not optimal, which increases the possibility of query failing.

Based on above all, we should adopt the distributed optimization when selection of the nodes executing the query. First, the sub queries are distributed to the responding data nodes. Second, the data nodes dynamically select the nodes executing the sub query according to location and speed of the current data nodes, location and speed of the other node storing the relation, size of relations and size of middle result. Thus, the data nodes form local sub query plan and execute it. At last, the middle result is submitted to the superior query. The superior query would process continuously until getting the last result and submit to the query node. The distributed algorithm could not only balance the load of mobile nodes but also dynamically select the executing query nodes according to current network situation in order to save energy.

This paper adopts query optimization based on heuristic rules. After the data nodes receive the sub queries from a query node, they optimize the sub queries according to the rules and dynamically select nodes to execute the query. The heuristic rules are as follows:

(1) The nodes where the middle result is should be close to the query node as much as possible.

(2) The nodes where the middle result is had better locate the center of the query node and the data node.

(3) When the distance between two data nodes is less than two hops, it selects one of the nodes as the node executing the sub query.

(4) When the distance between two data nodes is more than three hops and the distance becomes bigger and bigger, it selects another node to execute the sub query.

(5) When the distance between two data nodes is more than three hops and the distance becomes smaller and smaller, it selects one of the nodes as the node executing the sub query.

(6) Escaping node is a node far away the data nodes and query node and the distance between the escaping node and each data node is more than five hops. Also the distance between the escaping node and the query node is more than five hops. At this moment, it should process the escaping node firstly.

(7) It should not select the node with less energy to execute the sub query.

When applying the heuristic rules, we should estimate the variation tendency of distance among nodes and select third-party node to execute the sub query. Also, it needs to confirm and process the escaping nodes. Thus this paper proposes the estimation algorithm of distance variation tendency among nodes, the third-party node selection algorithm and the processing of escaping node algorithm. The query optimization algorithm is based on all three above algorithms. This section introduces these algorithms firstly, and then describes the optimization algorithm based on heuristic rules.

a) Estimation Algorithm of Distance Variation Tendency

Supposed each node could locate current location and speed of itself. First we assume the location and speed of nodes A and B at t_0. The location of A is (X_0,Y_0) and speed of A is (V_x,V_y). The location of B is (X_1,Y_1) and speed of A is (V'_x,V'_y). If A and B do not report new (location, speed), the location of A is $(X_0 + V_x *(t - t_0),\ Y_0 + V_y *(t - t_0))$ and the location of B is $(X_1 + V'_x *(t - t_0),\ Y_1 + V'_y *(t - t_0))$ at t and $t > t_0$.

According to the location of A and B at $t > t_0$, we use the distance formula of two points to compute the distance between A and B, which is $dis(t)=\sqrt{((X_0+V_x*(t-t_0))-(X_1+V'_x*(t-t_0)))^2+((Y_0+V_y*(t-t_0))-(Y_1+V'_y*(t-t_0)))^2}$. It could be changed into the form of $dis(t)=\sqrt{at^2+bt+c}$. a, b and c are constant numbers. Here

$$a=(V_x-V'_x)^2+(V_y-V'_y)^2,$$

$$b=2(V_x-V'_x)[(X_0-X_1)+(V'_x-V_x)t_0]+2(V_y-V'_y)[(Y_0-Y_1)+(V'_y-V_y)t_0],$$

$$c=[(X_0-X_1)+(V'_x-V_x)t_0]^2+[(Y_0-Y_1)+(V'_y-V_y)t_0]^2.$$

$dis(t)=\sqrt{at^2+bt+c}$ is a function about parameter t. The values of a, b and c are dependent on the location and speed of A and B at t_0. It is obvious that the distance of A and B is changing as time goes on. In order to get the variation tendency of distance between A and B, we make derivation of the function $dis(t)=\sqrt{at^2+bt+c}$. The derivation of dis(t) is $dis'(t)=\frac{2at+b}{\sqrt{at^2+bt+c}}$. $dis'(t)>0$ means dis(t) is increasing function at t moment, which indicates the distance between A and B is increasing. $dis'(t)<0$ means dis(t) is decreasing function at t moment, which indicates the distance between A and B is decreasing.

Thus the variation tendency could be confirmd with positive or negative of the derivation of distance between A and B. After current time is substituted into the formula, $2at + b > 0$ means the distance of two nodes is increasing, and $2at + b < 0$ means the distance is decreasing.

b) Third-party Node Selection Algorithm

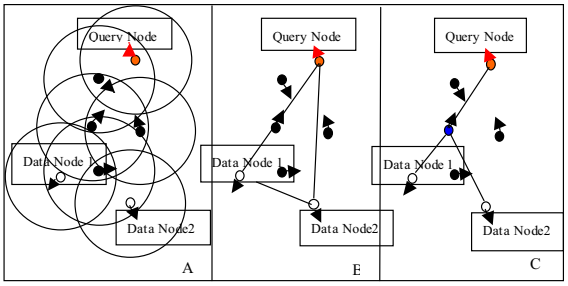

Fig.5 Selection of third-party node

As figure 5-A shows, the distance between two data nodes where the relations locate is more than three hops, and the distance between the two nodes and the query node is increasing. At this time, both the query node and one of the data nodes, no matter which node executes the current sub

query, the energy cost would be very large because the distance is far. In this instance, we should select a relay node in the center of the query node and the two data nodes, known as the third-party node. The sub query would be executed in the third-party node. After finish execution of the current sub query, the third-party node submits the result to the query node. The relay method could solve the energy cost problem caused by the long distance.

Next we introduce how to select the third-party node in detail. First, each data nodes where the sub query locates confirm whether the distance with the query node is more than three hops and the distance is increasing continuously. If it is true, the two data nodes broadcast the message towards the query node. When the mobile nodes between the data node and the query node receive the broadcasting message, they confirm the distance between themselves and the query node is less than two hops. If the distance is less than two hops, they confirm whether the distance with the query node is decreasing. If the distance is decreasing, the mobile nodes would be candidate nodes, and the location and speed of the nodes would be sent to the nodes where the sub query locates. When the nodes with the sub query receive the information, they would select the third-party node according to the follow steps.

1 To select the candidate node where other sub query locates priorly as the third-party node.
2 If there are many candidate nodes, it selects the one which the distance is decreasing most quickly as the third-party node.
3 If there is no node where other sub query locates in the optional nodes, it selects the one which the distance is decreasing most quickly as the third-party node.

Figure 5-B and 5-C show the procedure of selecting third-party node.

c) Processing of Escaping Node Algorithm

If the escaping node could not be processed in time, it will increase the energy cost of query. The escaping node would become isolated point if it is far away from the query node and other nodes, which results in the query failure. Thus it is important to process the escaping node in the second optimization phase.

Next we introduce the idea of processing escaping node. The data node N1 confirms whether the distance between itself and the query node QN1 is more than five hops firstly. If the distance is more than five hops, it confirms whether the distance is increasing continuously. If the distance is increasing, the data node N1 is escaping node. If the sub query S1 where N1 locates is independent of other sub queries, N1 is processed by the current sub query S1. If S1 is dependent on other sub queries, it would process N1 firstly without waiting for the information from other sub queries. At this moment, S1 sends information to another sub query S2 which S1 is dependent on. It confirms whether the data node N2 where S2 locates is escaping node. If the data node N2 is not escaping node and it has the same join attribute with N1, it processes the join operation between N1 and N2. If N2 is also an escaping node, N1 would broadcast its location and speed to all the data nodes and send its relation to the nearest data node which is not escaping node. Processing escaping node priorly could forward the effective data before the escaping node becomes isolated node, which contributes to improve the probability of success when executing a query.

D. Query Plan Execution Algorithm

Query plan execution algorithm executes from bottom to up according to the dependency relationship between queries. The main idea of the algorithm is as follows.

1 IF a sub query depends on other sub queries
2 THEN
2.1 Wait for messages from other sub queries, which means other sub queries has finished
2.2 IF receive SUB_QUERY_FINISH message
2.3 THEN
2.3.1 Run Query Execution Nodes Selection Algorithm to select nodes executing the query
2.3.2 Select an adaptable join algorithm[7] and form a local query plan
2.3.3 According to the local query plan to execute the sub queries
2.3.4 IF finish executing the sub query
2.3.5 THEN
Send SUB_QUERY_FINISH message to the higher sub query
2.3.6 END IF
2.4 END IF
3 END IF
4 IF the sub query does not depend on other sub queries
5 THEN
5.1 Run Query Execution Nodes Selection Algorithm to select nodes executing the query
5.2 Select an adaptable join algorithm and form a local query plan
5.3 According to the local query plan to execute the sub queries
5.4 IF finish executing the sub queries
5.5 THEN
Send SUB_QUERY_FINISH message to the higher sub query
5.6 END IF
6 END IF
7 IF the query node receives SUB_QUERY_FINISH message
8 THEN
8.1 Send QUERY_FINISH to data nodes executing the query
8.2 Receive the query result
8.3 RETURN
9 END IF

III. EXPERIMENT RESULTS AND ANALYSIS

All the experiments are carried out in a simulative mobile Ad Hoc network consisting of 10 nodes in the 1000*1000m^2 field. The speed of a node is between 1m/s and 20m/s and the covered range of each node is 200m. The cycle for updating position of a node is 1 second. That is, a node will move to a new place per 1 second. The sending power of a node is 13.5mW, and the receiving power is 10.5mW. The initial

energy is 95W which equals to the battery energy of a notebook computer.

We execute 4 groups of multi-join queries to evaluate two phases query optimization algorithm. Each relation is distributed on different nodes and the number of tuples in a relation is between 200 and 1000. Figure 6 shows the influence of the speed of mobile node over the energy cost of multi join query. When the node moves slowly, the energy cost of multi join query is more than the one when the node moves quickly. It is obvious that the energy costs in two situations are almost near. The reason is when selecting the nodes to execute the sub query, it limits the data nodes in a small region. At the same time, it takes account of the escaping nodes, which weakens the influence of speed of nodes over the energy cost of query.

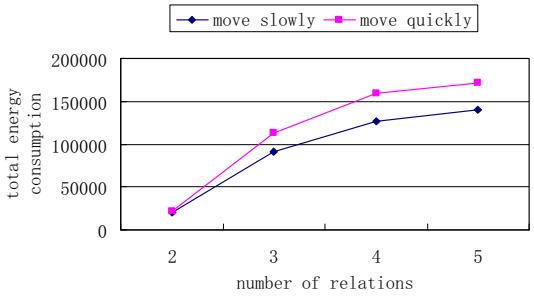

Fig.6 Relationship between movement speed of node and energy consumption

Figure 7 shows the influence of frequently changing of node moving direction over the energy cost of multi join query. When the number of relations is increasing and the nodes change direction frequently, the energy cost is higher. When the nodes move frequently, the selecting nodes algorithm would update the moving direction of the data nodes, which increases the energy cost of communication. Thus the total energy cost of join operation is increased.

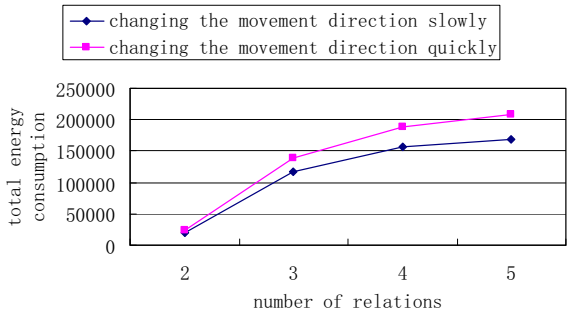

Fig.7 Relationship between speed of changing the movement direction and energy consumption

The experiments for the two phases optimization algorithm and centralized optimization algorithm are both conducted, which are according to the four testing plans of table I. The experiment results are shown in figure 8. The responding time of two phases optimization algorithm is obviously less than the one of centralized optimization algorithm. Because two phases algorithm greatly considers the parallelism among sub queries, the independent sub queries could executing parallely, which could effectively reduce the query execution time. Centralized optimization algorithm could not dynamically select the executing nodes and ignores the parallelism of join operation. Thus its query efficiency is not very well.

TABLE I. LIST OF TEST PLANS

	Number of query nodes	*Dataset(rows)*
Plan 1	2	5000
Plan 2	3	10000
Plan 3	4	15000
Plan 4	5	20000

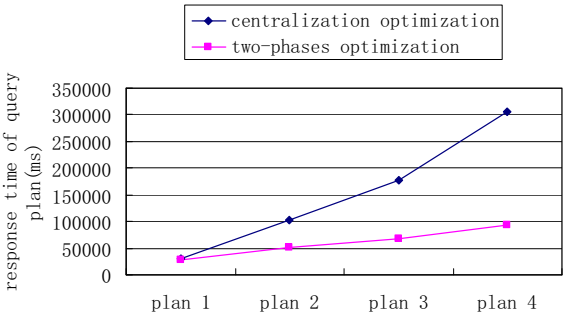

Fig.8 Comparison of performance of algorithms

IV. CONCLUSION

This paper takes the characteristics of the Ad Hoc network into account and designs the query representation model based on multi marked tree. The cost model based on energy and the two phases optimization algorithm are proposed. The two phases algorithm solves the problems of the inaccuracy of cost estimation, the high energy cost of query node and the great communication cost. At the same time, it completely develops the parallelism of sub queries, which optimizes the query execution time.

REFERENCES

[1] H. Kottkamp and O. Zukunft. Location-Aware Query Processing in Mobile Database Systems. In Proc. of the ACM Symposium on Applied Computing, Feb. 1998, pp. 416–423.

[2] Chang-Hung Lee, M. Chen: Using Remote Joins for the Processing of Distributed Mobile Queries. DASFAA 2001: pp. 226–233.

[3] Hüseyin Gökmen Gök, Özgür Ulusoy. Transmission of Continuous Query Results in Mobile Computing Systems. Information Sciences, Vol.125,No.1–4,Jun. 2000, pp. 37–63.

[4] Yongluan Zhou, Beng Chin Ooi, Kian-Lee Tan, Wee Hyong Tok. An Adaptable Distributed Query Processing Architecture. Data Knowledge Engineering 53(3), 2005, pp.283–309.

[5] Jinbao Li, Jianzhong Li. Query Processing in Mobile Ad Hoc Wireless Networks. International Conference on Computer and Information Technology, Sep. 2004, pp.633–638.

[6] Leslie D. Fife, Le Gruenwald. Research Issues for Data Communication in Mobile Ad-Hoc Network Database Systems. SIGMOD Record, Vol. 32, No. 2, Jun. 2003, pp.42–47.

[7] Eric Lo, Nikos Mamoulis, David Wai-Lok Cheung, Wai-Shing Ho, Panos Kalnis. Processing Ad-Hoc Joins on Mobile Devices. DEXA 2004: pp.611-621.