SCIENTIFIC RESEARCH CENTER
International Journal of Research in Science and Engineering
Volume 1, Issue 2

Reduce transfer costs allocation in distributed database unique pieces using genetic algorithms

Seyed Masoum Hoseini , [2*]Mina Mahdavi

[1]Department Of Computer Engineering , Amol Branch , Islamic Azad University , Amol , Iran

[*2]Department Of Computer Engineering , Amol Branch , Islamic Azad University , Amol , Iran

Abstract

Distributed database were developed to respond to the needs of distributed computing. Unlike traditional database systems, distributed database distributed at multiple sites. A distributed database is a collection of sites that are connected over the network, each of the sites, Have their own database, But they can work together, so that each user in each site can have access to all the data in the network, just like that all user data is stored on the site. The main objective of the allocation algorithm to determine the proportion of the various parts of the site is to reduce the cost of shipping the piece. This study presents an algorithm to reduce the cost of data transfer when allocating parts is, in such a manner that First of all, parts of the of the site, it should be, second reduce the cost of transmission components sites should Third update costs for all components that are present at the site should be optimized, However, all the conditions for the transfer of part of the site is selected, so that the reliability and availability of the site increases. In this study, the assignment of parts to a site in a distributed database system is done using genetic algorithms, Efforts to reduce the cost of data transmission. To review and consider the advantages of the previous method, the initial population, we have tried to better optimality of previous approaches is the initial population And a genetic algorithm to select the best chromosome instead of two parameters, the three parameters used. Analytical study shows that the proposed algorithm to other algorithms, the benefits and availability of data and reduce costs in the allocation of parts.

Key words: Distributed database, data segmentation, GA

1 – Introduction

Advances in networking and database technology in recent decades has led to the development of distributed database systems. A distributed database is a collection of nodes that are connected to the network, each of which has its own database From site, they can work together, Therefore, each user at each site can have access to all the data on the network's Web site just like that all the data is stored. Distributed database in order to achieve the purpose of the allocation of plots used.

Single fragment could be a file in the file allocation will be subject to the same allocation problem is NP allocation is a matter of degree. It requires fast heuristics to produce effective solutions. In addition, the optimal allocation of database components to severe, depending on the strategy used by one of the components of the genetic algorithm is optimal allocation strategies. The use of evolutionary algorithms in optimization problems, a new approach has been considerable interest in the last few years and has shown good efficacy compared to the old methods.

2- Background And the Literature Research

- Literature Research

Segmentation data:

Design stage of distribution, including the distribution and allocation of the relationship between the parts. The purpose of segmentation to better distribution. The assignment is to put the best parts of the site in the database is very important. Care must be taken in placing redundant parts to maintain consistency and efficiency available.

genetic algorithm

A genetic algorithm-based search technique applicable laws and mechanisms of natural selection and survival of the most competent, the evolution is natural. By simulating the natural evolution, a genetic algorithm can effectively search for and scope of the problem difficult to solve simple problems. In addition to imitate the biological selection and amplification techniques, a genetic algorithm can perform a search in a general and independent. A GA to solve the problem of a very large number of possible solutions to be generated. Each of these solutions is evaluated using a fitness function. Then some of the best solutions to generate new solutions, which makes this work is the development of solutions.

- Literature

- ✓ Mr. Sovin, Sarab in 2011, a genetic algorithm as a technique to obtain a near optimal solution to the problem is presented, Identify the duplicate data can also be stored on other sites This increases the storage space and low productivity.
- ✓ Mr. Basda and colleagues in 2006, consistent vertical partitioning methods have introduced a distributed system And Mr. Huang and colleagues (2001) proposed a method for grouping components based on the size of each piece And the sharing of

parts among each group sites And how to move the pieces around to sites close together to solve these issues have been addressed.

3 - Analysis of the data

Schema chromosomes in genetic algorithms:

Site Number of fragment 1	Site Number of fragment 2	Site Number of fragment 3	. . .	Site Number of fragment n-2	Site Number of fragment n-1	Site Number of fragment n

Figure 1: Structure of chromosomes

3-1- Initialization:

```
private void SizeNode()
{
    for (int i = 0; i < 50; i++)
    {
        ArrSizeFragment[i] = Rand.Next(500, 1023);/
    }
    for (int j = 0; j < 10; j++)
    {
        ArrSizeNode[j] = (Rand.Next(5, 100) * 1024);//
    }
    for (int i = 0; i < NumberNode; i++)/
    {
        for (int j = i + 1; j < NumberNode; j++)
        {
            CostTransmission[i, j] = Rand.Next(50, 100);//
        }
    }

}
```

3-2-Function requirements

```
public void Required(int SelectNode, int Numbernode)
{
    int SelectFragment = 0;
    if (SelectNode == 1)
    {
        SelectFragment = Rand.Next(0, 50);//
        if (ArrAfterSizeNode[Numbernode] >= ArrSizeFragment[SelectFragment])//
        {
            if (SplitArrayHnode(SelectFragment, Numbernode))//
            {
                ArrRequird[Numbernode, SelectFragment] = 1;

                string MeghdarGhabli = ArrayHNode[SelectFragment];
                ArrayHNode[SelectFragment] = MeghdarGhabli + "," + Numbernode;
                OptimalAllocation(SelectFragment, Numbernode);
                //textBox3.Text += ArrayHNode[SelectFragment];
                //textBox4.Text += Numbernode + "," + SelectFragment + "   ";
            }
            else
            {
                Required(SelectNode, Numbernode);
            }
        }

    }
}
```

3-3-The initial population

```
public void random1()// ☻ Operation ENCODING ☻
{
    bool b;
    int j = 0, n = NumberFragment - 2;
    int i = Rand.Next(0, 50);
    ArrRandAllocation[j] = i;
    // textBox10.Text += i + "  ";
    while (j <= n)
    {
        i = Rand.Next(0, 50);
        b = false;
        for (int k = 0; k <= j; k++)
        {
            if (ArrRandAllocation[k] == i)
                b = true;
        }
        if (b == false)
        {
            j += 1;
            ArrRandAllocation[j] = i;
            //  textBox10.Text += i + "  ";
        }
    }
}
```

```
private void population()//
{
    for (int i = 0; i < NP; i++)
    {
        random1();
        TachsisAval1();
        TakhsisAval();//
        StartTakhsis();
        CopyArrToCHromosome(i);
        ClearAllArray();
    }

}


  public void TachsisAval1()//
  {
      bool b;
      int j = 0, n = NumberFragment - 2;
      int i = Rand.Next(0, 50);
      ArrRandAllocationT[j] = i;
      //  textBox1.Text += i + "  ";
      while (j <= n)
      {
          i = Rand.Next(0, 50);
          b = false;
          for (int k = 0; k <= j; k++)
          {
              if (ArrRandAllocationT[k] == i)
                  b = true;
          }
          if (b == false)
          {
              j += 1;
              ArrRandAllocationT[j] = i;
              // textBox1.Text += i + "  ";
          }
      }
  }
    private void StartTakhsis()
    {
        int SelectNode = 0;
        for (int j = 0; j < 5; j++)
        {
            for (int i = 0; i < NumberNode; i++)
            {
                SelectNode = Rand.Next(0, 2);
                Required(SelectNode, i);
            }
        }
    }
```

3-4-Combine function

```
for (int i = NP; i < NP + NC; i++)//☻ Operation CROSSOVER ☻

    int Fragment = 0, CHromosomeFirst = 0, CHromosomeSecond = 0;
    Fragment = Rand.Next(15, NumberFragment);
    CHromosomeFirst = Rand.Next(0, NumberFragment);
    CHromosomeSecond = Rand.Next(0, NumberFragment);
    crossover(Fragment, CHromosomeFirst, CHromosomeSecond, i);
}

private void crossover(int Frag, int CHF, int CHS, int NumberHomeArray)
{
    for (int i = 0; i < Frag; i++)
    {
        ArrCHromosome[NumberHomeArray, i] = ArrCHromosome[CHF, i];
    }
    for (int i = Frag; i < NumberFragment; i++)
    {
        ArrCHromosome[NumberHomeArray, i] = ArrCHromosome[CHS, i];
    }

}
```

3-5-Function mutation

```
for (int i = NP + NC; i < NP + NM + NC; i++)//☻ Operation MUTATION ☻
{
    int Fragment = 0, CHromosome = 0;
    Fragment = Rand.Next(0, NumberFragment);
    CHromosome = Rand.Next(0, NP);
    mutation(CHromosome, Fragment, i);
}
private void mutation(int CHromosome, int Fragment, int NumberHomeArray)
{
    for (int i = 0; i < NumberFragment; i++)
    {
        if (i == Fragment)
        {
            ArrCHromosome[NumberHomeArray, i] = "1";
        }
        else
        {
            ArrCHromosome[NumberHomeArray, i] = ArrCHromosome[CHromosome, i];
        }

    }
}
```

3-6-Fitness function

```
private void OptimalAllocation(int SelFra, int NumNode)
{

    string s = string.Empty;
    string[] x = new string[10];
    s = string.Empty;
    s = ArrayHNode[SelFra];
    x = s.Split(',');
    int Cost = 0, Row = 0, Column = NumNode, MainCost = 100;
    if (x.Length > 1)
    {
        for (int i = 0; i < x.Length; i++)
        {
            Row = Convert.ToInt32(x[i]);
            Cost = 0;
            if (Row > Column)
            {
                Column = Convert.ToInt32(x[i]);
                Row = NumNode;
            }
            Cost = CostTransmission[Row, Column];
            if (MainCost > Cost)
            {
                MainCost = Cost;
            }
        }
        CostAll += MainCost;
    }
}
```

4- Results

In this study, an allocation algorithm called genetic algorithms were tested components. This algorithm is based on a threshold algorithm, but different strategies for transferring data to other sites using the parts. For the experiments, the cost factor in the transmission network, which is considered one of the parameters The effect of these parameters on different conditions were studied. Results show that the proposed algorithm is compared to a threshold and greedily transfer fee and a great improvement can be achieved using this algorithm.

References

1- Rouhani Rankoohi A. 2009. Anewant colony optimization based algorithm for data allocation problem in distributed databases. J Knowl Inf Syst (Springer)

2- Huang Y. F. And Chen J. H. 2001. Fragment Allocation inDistributed Database Design , Journal of Information Science and Engineering 17, 491-506.

3- Ulus T and Uysal M. 2003. Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems, Pakistan Journal of Information and Technology 2 (3): 231-239.

4- Chu W, Fellow I, Ieong T. 1993. A transaction-based approach to vertical partitioning for relational database systems. IEEE Trans Softw Eng 19(8):804–8012.

5- Cui X, Potok TE, Palathingal P. 2005. Document clustering using particle swarm optimization. In: IEEE transaction on swarm intelligence symposium(SIS) proceedings, pp 185–191.

6- Du J, Alhajj R, Barker K. 2006. Genetic algorithms based approach to database vertical partition. J Intell Inf Syst 26:167–183.

7- Eisner M, Severance D. 1976. Mathematical techniques for efficient record segmentation in large shared databases. J ACM 23(4.(

8- Hababeh I. O. 2005. A Method for Fragment Allocation Design in the Distributed Database Systems, The Sixth Annual U.A.E. University Research Conference.

9- Baseda R, Tasharofi S. Rahgozar M. 2006. Near Neighborhood Allocation: A Novel Dynamic Data Allocation Algorithm in DDB.