第 8 卷第 2 期
1997 年 2 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 8, No. 2
Feb. 1997

利用遗传算法求解文件分配问题*

孟祥武 程 虎

(中国科学院软件研究所 北京 100080)

摘要 文件分配问题是计算机网络和分布式系统中一个非常重要的问题. 本文提出一种用遗传算法求解文件分配问题的新方法,该方法能较好地解决工程中的文件优化分配问题,还可以应用到其它资源需要分配的领域.

关键词 文件分配问题,遗传算法,人工智能,计算机网络.

在总线型微机局域网环境下,进行网络数据库设计或分布式系统设计时,就会遇到服务器上的文件分配问题. 作者在研制开发几个管理信息与决策支持微机网络系统时,都遇到了这个问题. 这个问题最早是在 1969 年被 W. W. CHU 首先提出的[1],以后许多学者进行了研究,发表了大量的论文. 1982 年 L. W. DOWDY 和 D. V. FOSTER 写了一篇著名的综述性文章[2],评价了已存在的各种方法,指出了它们的优缺点,提出还需要进一步的研究. 近几年这方面的研究工作又有了进展[3~5],但这些方法相对而言比较复杂,难以实现,缺乏应用价值. 所以在具体应用中,常常用到一些分配原则,如多台服务器尽量分配数量大致相同的工作站和文件,每个工作站的文件最好分配在同一个服务器上.[4]根据这些原则,由人来决定文件分配问题. 本文研究用遗传算法求解文件分配问题,方法简单,易于实现.

1 文件分配问题

在实际工程问题中,服务器往往成为整个网络系统的"瓶颈",所以提高服务器的利用率是提高整个网络性能的关键. 实现文件的优化分配,使每个服务器的负荷较为平均,减少访问冲突,这样就能极大地提高网络系统中服务器的利用率,进而提高整个网络系统的性能.

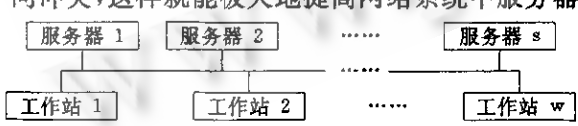

图1 总线型微机局域网

这里设有 $n+1$ 个文件要放在 s 个服务器上,服务器上文件 $F_i(0 \leqslant i \leqslant n)$ 被网络中某个工作站访问的概率为 P_i,并且服务器上 $n+1$ 个文件被访问的事件为相互独立事件. 需将这 $n+1$ 个文件合理地分配到 s 个服务器上,还能方便地动态改变文件及其拷贝在各台服务器上的分布,这里文件包括数据文件和程序文件.

---

* 作者孟祥武,1966 年生,博士生,讲师,主要研究领域为遗传算法,神经网络和人工智能. 程虎,1938 年生,博士导师,研究员,主要研究领域为语言编译,软件工程,神经网络和人工智能.

本文通讯联系人:孟祥武,北京 100080,中国科学院软件研究所

本文 1996-01-15 收到修改稿

2 分配策略

2.1 存储优化分配策略

文件分配问题涉及的因素很多,主要有被访问的时间代价(即被访问的次数)和存储代价.下面介绍我们的存储分配解决方法.

根据软件工程原理:每个文件不宜太大,应该模块化.在实际应用中,程序文件不宜太大,也不宜太小.数据文件也一样,因文件太大,查询、更新、传输、代价都比较大,况且一个数据文件太大,就会出现很多问题,例如:大数据文件,往往是字典性质的,许多工作站用户都要用,所以访问大文件时间长,这样并发控制问题就更加复杂化了,访问冲突率也更高了,降低了文件的使用效率.另外,大数据文件在存储、更新、传输等操作过程中,出错的概率也相对大起来了,降低了可靠性.所以程序文件和数据文件都不宜太大.这也是从工程实际中得出的经验.

这里讲的文件是具体的物理文件,不是逻辑文件.一个逻辑文件可由几个物理文件组成.物理文件是面向机器的,而逻辑文件是面向用户的,因而逻辑文件可较大,不受此限制.

所以我们这里文件大小差别不是很大,故每个服务器上分配大致相等的文件数,结果就是每个服务器上被占用的存储量相似.几方面的研究结果也表明:每个服务器上最好分配大致相等的文件数.[4]这里有个条件,所要分配到服务器上的文件存储容量之和要小于所有服务器的容量之和.若服务器上的容量不够,首先考虑删除多拷贝(多副本)文件的一个或几个拷贝,或将使用率低的文件压缩存储以节省存储空间.本文暂不考虑这个问题.

我们用每个服务器上分配大致相同的文件数来解决文件的存储优化分配问题.

2.2 时间代价与存储代价的统一分配策略

对于被访问概率与其它多数文件被访问概率相似的文件,或被访问概率很低的文件,仅该文件参加分配.若文件 $F_t(0\leqslant t\leqslant n)$ 被访问的概率大大高于其它文件的被访问概率,这说明文件 F_t 经常被使用,因此我们可生成文件 F_t 的一个或几个拷贝(副本).这里具体根据拷贝数 C 来决定是否让文件 F_t 有拷贝,若有,有几个拷贝?

$$C=\lfloor P_i/P_f\rfloor-1;$$

$\lfloor i\rfloor$ 表示取小于等于 i 的整数.P_i 为文件 F_t 的被访问概率.

$$P_f=(P_0+P_1+\cdots+P_n)/(n+1).$$

若 $C\geqslant 1$,就应有拷贝,其拷贝数为 C.若 $C<1$,就无拷贝.

对于文件 F_t 及其拷贝,有 2 种分配方法:(1)文件 F_t 的拷贝可以分配到经常访问它的工作站上,若仅有个别工作站常访问它.(2)文件 F_t 的拷贝也分配到服务器上,若多个工作站经常访问它.

一般的分配原则是:若经常访问 F_t 的工作站数目明显高于 F_t 的拷贝数 C,则将 F_t 的拷贝分配到服务器上;反之则将 F_t 的拷贝分配到工作站上.

当文件 F_t 的拷贝要分配到服务器上时,即要分配到服务器上的文件增加了,此时其拷贝对应的文件名为:$F_{n+1},F_{n+2},\cdots,F_{n+c}$.

最后,$n=n+c$.

这里又出现了另一个问题,局域网及分布式系统中多拷贝的一致性更新问题,这个问题已有一些解决办法.[6,7]还需要进一步的研究,这也是一个很有意义的研究课题.

我们这里采用多拷贝技术来解决系统中某些文件被经常使用的现象,以此来减少访问冲突.不仅每个服务器上被访问的概率大致相同,而且每个服务器上分配大致相同的文件数,使被访问文件的时间代价与存储代价统一起来,因而优化分配结果更加合理.

3 遗传算法求解方法

3.1 编码方法(Encoding)

有 $n+1$ 个文件,每个文件依次表示为 $F_0,F_1,\cdots,F_n$,相应被访问的概率为 $P_0,P_1,\cdots,P_n$. 对这 $n+1$ 个文件,我们用一个 $n+1$ 位的二进制串来表示,若该串的第 0 位为 1,则表示文件 F_0 在服务器上;反之不在服务器上. 即:

$$b_i=\begin{cases}1, & \text{文件 } F_i \text{ 在服务器上;}\\ 0, & \text{文件 } F_i \text{ 不在服务器上}\end{cases}\qquad (0\leqslant i\leqslant n)$$

每次将一个服务器分配完后,再继续给下一个服务器分配,因为此时文件数变少了,所以我们用一个较短的二进制串来表示剩余文件,串长为剩余文件数目.

原二进制串中为 0 的位所代表的文件就是这次的剩余文件,剩余文件重新形成序列,文件名为 $F_0,F_1,\cdots,F_L$,其相应的被访问概率为 $P_0,P_1,\cdots,P_L$;L 为剩余文件数,$0\leqslant L\leqslant n$.

即每分配一个服务器,二进制串根据情况变化一次,以此来克服遗传算法中传统的固定长度二进制串的缺点.[8,9]编码串是动态的.

3.2 初始化群体

用过程 Initialize 来实现,方法是随机生成初始化的串群体. 在串群体中,串长度都是相同的,串长为需要分配的文件数. 群体的大小根据需要(要求的分配时间等),按经验或实验给出. 分布均匀的二进制串能使算法更加有效.

3.3 选择(Selection)

借用达尔文的生物进化论中的自然选择(Natural Selection)思想,按照"适者生存"的原则对串进行复制. 用适应度函数计算每个串的适应度,选择适应度高的串,生成下一代,去掉适应度差的串.

3.4 交换(Crossover)

交换是 2 个串按照一定的概率(交换概率 Pc)从某一位开始逐位互换. 这里先在串群体中,随机地选择 2 个串,成为一对串,变成多对串后,对每对串随机的选择一个交换点,例如串长为 $n+1$,则可选择一整数 i,$0\leqslant i\leqslant n$,i 为交换点,对 2 个串从第 0 位到第 i 位进行互换,形成 2 个新串. 是否发生交换操作,还要受交换概率的控制,选择好一对串后,在 0、1 之间产生一个随机数,若该随机数大于 Pc 则发生交换,否则保持原状. Pc 也是根据经验或实验确定,一般可为 0.5 左右.

3.5 突变(Mutation)

二进制串的某一位按照一定的概率(突变概率 Pm)发生反转,0 变 1,1 变 0. 这里 Pm 较小,Pm 可小于 0.001.

3.6 适应度函数(Fitness Function)

这里我们用 Evalution 过程来实现. 上述每个串,串中为 1 的位,表示该位对应的文件应在该服务器上.

有 $n+1$ 个文件 $F_0,F_1,\cdots,F_n$,其相应被工作站访问的概率为 $P_0,P_1,\cdots,P_n$.

令 P_f 为　　$P_f=(P_0+P_1+\cdots+P_n)/(n+1)$

这 $n+1$ 个文件要被平均分配到 s 个服务器上,故每个服务器应有 m 或 $m+1$ 个文件.

$$m=\lfloor (n+1)/s \rfloor$$

因为服务器上 $n+1$ 个文件被访问为相互独立事件,所以每个服务器被访问的概率应在 Pmin 与 Pmax 之间:

$$Pmin=\sum_{i=1}^{m}Q_i-\sum_{i<j=2}^{m}Q_i\cdot Q_j+\sum_{i<j<k=3}^{m}Q_i\cdot Q_j\cdot Q_k+\cdots+(-1)^{m-1}\cdot Q_1\cdot Q_2\cdot\cdots\cdot Q_m,$$

$$Pmax=\sum_{i=1}^{m+1}Q_i-\sum_{i<j=2}^{m+1}Q_i\cdot Q_j+\sum_{i<j<k=3}^{m+1}Q_i\cdot Q_j\cdot Q_k+\cdots+(-1)^{m}\cdot Q_1\cdot Q_2\cdot\cdots\cdot Q_{m+1},$$

这里 $Q_1=Q_2=\cdots=Q_m=Q_{m+1}=P_f$.

按照二进制串计算相应服务器被访问的概率 Pe.

设二进制串长为 $L(L>0)$,串中含 1 的个数为 t.

当 t 大于 0,小于等于要分配的文件数时

$$Pe=\sum_{i=1}^{L}P_i\cdot b_i-\sum_{i<j=2}^{L}P_i\cdot P_j\cdot b_i\cdot b_j+\sum_{i<j<k=3}^{L}P_i\cdot P_j\cdot P_k\cdot b_i\cdot b_j\cdot b_k+\cdots$$
$$+(-1)^{L-1}\cdot P_1\cdot P_2\cdot\cdots\cdot P_L\cdot b_1\cdot b_2\cdot\cdots\cdot b_L$$

$$b_i=\begin{cases}1, & \text{文件 } F_i \text{ 在服务器上;}\\ 0, & \text{文件 } F_i \text{ 不在服务器上}\end{cases}\qquad (0\leqslant i\leqslant L)$$

反之,$Pe=0$.

根据 Pe 与 Pmin 和 Pmax 之间的差来计算适应度,当 $Pmin\leqslant Pe\leqslant Pmax$ 时,适应度最高,算法结束时串所代表的文件就是要被分配到服务器上的文件. 所有服务器分配完毕,求解文件分配问题结束.

3.7 相应的遗传算法描述

```
Procedure GA Program
  Begin
    For i=1 to s-1 do
      Begin
        Initialize;
        Evaluation;
        While (not termination-condition) do
          Begin
            Selection;
            Crossover;
            Mutation;
            Evaluation
          End
      End
  End.
```

4 试验结果

对于已知的被分配文件集合，可以用被使用频率来近似估算其概率. 这里又可以用相对频率来代替概率.

因一般不同的文件，有不同的被使用频率. 特别是数据文件，如年报、季报、月报、旬报、周报和日报，所以可得到每个文件的近似使用频率，另外通过网络监控程序也可进一步得出精确的使用频率.

在实际中，有许多随机因素影响文件的使用频率，但在一段较长时期内，每个文件还是有一个较为准确的被使用频率.

算法中的参数值分别为 $POP\ SIZE=10$; $Pc=0.6$; $Pm=0.05$. 该算法已在 486 微机上用C 语言实现. 下面是一个例子.

有一个文件集合 $F_0,F_1,\cdots,F_{10}$，其相应被工作站访问的概率为 $P_0,P_1,\cdots,P_{10}$，对应的值是 0.60,0.11,0.56,0.08,0.31,0.04,0.20,0.11,0.13,0.18,0.49. 要被分配到 3 个服务器上.

程序运行结果：第 1 次分配文件 F_1,F_8,F_9,F_{10}到第 1 个服务器上，第 2 次分配文件 F_0, F_3,F_5,F_6 到第 2 个服务器上，最后将剩余文件 F_2,F_4,F_7 分配到第 3 个服务器上.

5 结束语

与其它方法相比较[1~5]，本文提出的方法有以下几个优点：(1)每个文件被访问具有一定的规律和随机性，不确定性. 为了处理这种特性引入了概率分析. (2)需要的有关数据很少且易得到，仅需文件被访问概率及大小. 而其它方法，需要许多相关数据，在实际工程中，有时那些所需数据是无法或难以得到的. 本方法需要的数据，用户很容易提供. (3)因数据少，易得到，所以也易被修改，故很适用于变化的环境. (4)能得到多个解，即可得到多个分配方案可供选择.

文件分配问题常常出现在实际应用中，本文尝试利用近年兴起的遗传算法来求解它. 结果表明遗传算法能较好地解决上述问题，算法简单，易于实现. 此方法可以方便地动态改变文件及其拷贝在各台服务器上的分布，是一种很实用的新方法. 该方法还可以应用到其它领域，解决资源合理分配的问题.

参考文献

1 Wesley W Chu. Optimal file allocation in a multiple computer system. IEEE Trans. on Computers. 1969, **C-18 (10)**: 885~889.

2 Lawrence W Dowdy, Derrell V Foster. Comparative models of the file assignment problem. ACM Computing Surveys. 1982, **14(2)**: 287~313.

3 Laurence J Laning, Michael S Leonard. File allocation in a distributed computer communication network. IEEE Trans. on Computers, 1983, **C-32(3)**: 232~244.

4 Woodside C Murray, Tripathi Satish K. Optimal allocation of file servers in a local network environment. IEEE Trans. on Software Eng., 1986, **SE-12(8)**: 844~848.

5 Murthy I, Ghosh D. File allocation involving worst case response times and link capacities: model and solution

procedure. European Journal of Operational Research, 1993,67(3):418～427.

6 Mukesh Singhal. A fully-distributed approach to concurrency control in replicated database systems. In: Proceedings of the Twelfth Annual International Computer Software & Applications Conference, Chicago, 1988. 353～360.

7 Jajodia S, Mutchler D. A pessimistic consistency control algorithm for replicated files which achieves high availability. IEEE Trans. on Software Eng., 1989,15(1):39～46.

8 Zurada Jacek M, Mark I Robert J, Robinson Charles J. Computational intelligence: imitating life. New York: IEEE Press, 1994.

9 Holland John H. Genetic algorithms. Scientific American, July 1992. 66～72.

SOLVING FILE ALLOCATION PROBLEM BASED ON GENETIC ALGORITHMS

MENG Xiangwu　CHENG Hu

(*Institute of Software　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　FAP (file allocation problem) is a very important problem in computer network and distributed system. This paper presented a new method to solve FAP based on genetic algorithms. The method can well solve optimal file allocation problem in engineering. The method can be also applied in other situations besides the FAP, in those situations resources need to be allocated.

Key words　File allocation problem, genetic algorithms, artificial intelligence, computer network.