УДК 681.3(06)

Ю.В. Ладыженский, Д.В. Бельков

# ДИНАМИЧЕСКОЕ РАЗМЕЩЕНИЕ ФАЙЛОВ ПО УЗЛАМ РАСПРЕДЕЛЕННОЙ САПР

Быстрое развитие информационных и коммуникационных технологий привело в настоящее время к возможности широкого распространения распределенных архитектур групповой обработки данных. Средством групповой обработки может быть многопользовательский интерфейс, реализованный на основе локальной вычислительной сети. Технология групповой обработки хорошо подходит для многих видов интеллектуальной коллективной работы, в том числе для проектирования сложных технических объектов.

Наиболее подходящим построением САПР может быть объектно-ориентированная машина распределенной архитектуры [1]. Частью машины САПР является собственная распределенная операционная система. В эту систему входит оптимизатор, который производит самоадаптацию размещения файлов в зависимости от интенсивности их использования. Оптимизация интенсивности информационного трафика между узлами вычислительной сети улучшает время ответа на запросы к файлам распределенной САПР.

В процессе длительной работы распределенной системы исходное размещение файлов может стать не оптимальным, поэтому план размещения файлов должен корректироваться. При динамическом размещении файлов по узлам вычислительной сети циклически выполняются два этапа: 1) определение интервала между перераспределениями файлов; 2) решение задачи перераспределения файлов.

При функционировании распределенной системы в каждом узле образуются 2 типа запросов: сетевой запрос, для обработки которого необходим файл, не содержащийся в том узле, где возник запрос; локальный запрос, для обработки которого необходим файл, содержащийся в том узле, где возник запрос. Обозначим: $F_{ij}$ - количество запросов к файлу i из узла j в единицу времени; $X_{ij}$ =1, если файл i

расположен в узле j, иначе $X_{ij}$=0; $V_i$ - объем файла i; $B_j$ - объем узла j, i=1...m, j=1…n. Значения $F_{ij}$ случайны и зависят от времени t, остальные величины постоянны. Обозначим: $L = \sum_{i=1}^{m}\sum_{j=1}^{n} F_{ij} X_{ij} V_i / B_j$ - суммарный поток локальных запросов; $Y_{ij} = 1 - X_{ij}$; $N = \sum_{i=1}^{m}\sum_{j=1}^{n} F_{ij} Y_{ij} V_i / B_j$ - суммарный поток сетевых запросов.

В режиме с L<N время реакции системы на запросы больше, чем в режиме с L>N за счет межпроцессорных взаимодействий, поэтому считаем, что в этом случае распределение файлов неудовлетворительно. Для оценки качества распределения файлов введем оценочную функцию системы: U = (L-N)/L = 1-N/L, если L ≥ N, иначе U=0. Если L ≥ N, N>0, то $0 \le U < 1$.

Зададим при k-ом перераспределении файлов шаг по времени - $T_p^k$: k=1,2,… , $T_p^k = T_p$, $T_p = \max_j T_j$, $T_j$ - время использования файлов узла j, заданное при k-ом перераспределении файлов. В течение интервала $T_p^k$ сеть функционирует от k-ого перераспределения файлов до проверки необходимости нового распределения файлов.

Поиск интервала между перераспределениями файлов состоит в следующем. Пусть в момент времени $t_k$ завершено k-ое перераспределение файлов и получены значения $L_k$, $N_k$, $U_k$. Установим счетчик шагов: s:=1. Сделаем шаг по времени длиной $T_p^k$ из точки с координатами (0, $t_k$) в точку с координатами (0, $t_{k+1}$). Если $U_{k+1}$≠0, то делаем еще один шаг длиной $T_p^k$ и наращиваем счетчик шагов: s:=s+1. Если $U_{k+1}$=0, то в момент времени $t_{k+1}$ необходимо перераспределение файлов. Таким образом, интервал между перераспределениями файлов равен сумме $\sum_{k=1}^{s} T_p^k$.

Поиск интервала между перераспределениями файлов показан на рисунке 1. Делаем шаг $T_p^1$ и проверяем условие $U_1 \neq 0$. Оно не выполняется. Поэтому необходимо перераспределение файлов, для того, чтобы обеспечить $U \neq 0$. После перераспределения делаем шаг $T_p^2$ и проверяем условие $U_2 \neq 0$. Оно выполняется. Поэтому делаем шаг $T_p^3 = T_p^2$ и проверяем условие $U_3 \neq 0$. Оно не выполняется. Поэтому необходимо перераспределение файлов, для того, чтобы обеспечить $U \neq 0$. После перераспределения делаем шаг $T_p^4$ и проверяем условие $U_4 \neq 0$. Оно выполняется. Поэтому в дальнейшем нужно сделать шаг $T_p^4 = T_p^5$ и т.д.

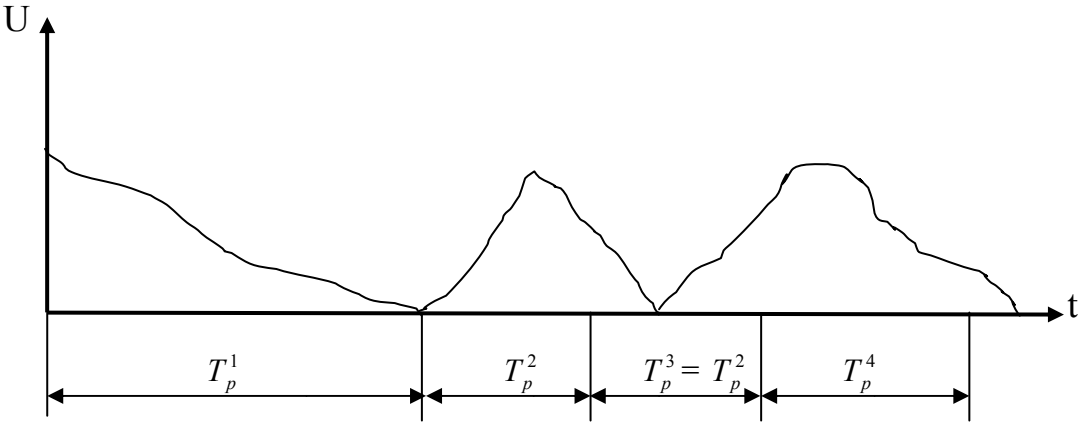


Рисунок 1 - Поиск интервала между перераспределениями файлов

Чтобы избежать излишних пересылок при перераспределении файлов, его целесообразно осуществлять с некоторым запаздыванием. Определение величины этого запаздывания представляет собой отдельную задачу.

Обозначим: $T_j$ - время использования файлов узла j; $\tau_j$ - допустимое время выполнения запроса к файлам узла j; $\rho_j$ - коэффициент загрузки узла j: $\rho_j = \sum_{i=1}^{m} F_{i,j} \cdot \tau_j$. Если $0 < \rho_j < 1$, то в узле j нет очереди запросов к файлам. Если $\rho_j > 1$, то в узле j возникает очередь запросов к файлам. Поэтому в случае $\rho_j > 1$ время реакции узла на запрос больше, чем в случае $0 < \rho_j < 1$. Для эффективной работы системы необходимо выполнение условий: $\rho_j \to \max$; $\rho_j < 1$, $T_j \to \max$,

j=1…n. Поэтому файлы должны быть размещены по узлам с целью максимизации функции (1) при ограничениях (2)-(4).

Целевая функция:

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\rho_j X_{ij} T_j V_i / B_j = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{i,j} X_{i,j} \to \max \quad (1)$$

Ограничения:

$$X_{ij} = \{0;1\}, \sum_{j=1}^{n} X_{ij} = 1 \quad (2)$$

$$\sum_{i=1}^{m} V_i X_{ij} \le B_j \quad (3)$$

$\rho_j$<1, j=1…n (4)

В задаче (1)-(4) максимизируется суммарное время использования файлов узлов. В ней необходимо найти матрицу размещений файлов X. Каждый файл размещается только в один узел. В задаче максимизируются $\rho_j$ и $T_j$, поэтому при k-ом перераспределении файлов величины $L_k$, $T_p^k$, $U_k$ стремятся к наибольшим значениям, а величина $N_k$ стремится к наименьшему значению. Таким образом, файлы размещаются с учетом динамики функционирования системы во времени. Задача относится к классу NP-трудных, так как содержит в качестве частного случая многомерную задачу о рюкзаке [2].

Эвристический алгоритм решения задачи (1.1)-(1.4) размещает файлы по принципу “в первый подходящий узел в порядке убывания размеров файлов” (FFD). Перед применением алгоритма файлы нужно упорядочить по убыванию их размеров, а узлы – по возрастанию объемов.

Для решения задачи (1)-(4):

1. Формируем матрицу C с элементами $C_{ij} = \rho_j T_j V_i / B_j$;
2. Задаем нулевые значения матрице размещения файлов X и матрице запретов размещения E;
3. Задаем нулевые значения вектору W;

4. Изменяем матрицу C: если $V_i > B_j$, то $C_{ij} := 0$;
5. Полагаем max = $C_{i1}$ и p=1.
6. Среди не запрещенных узлов ($E_{ij} \neq 1$) для каждого файла i находим $C_{ip} = \max C_{ij}$, j=1...n;
7. Если $W_p + V_i \leq B_p$, то размещаем его в узле p: $X_{ip} := 1$; $W_p := W_p + V_i$. Если $W_p + V_i > B_p$, то: а) запрещаем узел p: $E_{ip} := 1$ так как файл i нельзя поместить в узел j; б) среди всех узлов находим первый, не запрещенный узел k; в) полагаем p=k и max= $C_{ip}$; г) переходим к пункту 6.
8. Используя сформированную матрицу X, вычисляем функцию цели по формуле (1).

На этапе 7б может оказаться, что все узлы запрещены для размещения файла i, т.е. алгоритм не может решить задачу при заданных начальных условиях. В этом случае нужно увеличить объем узлов, а затем повторить вычисления с шага 1 [3].

Обозначим: max – значение максимального элемента i-ой строки матрицы C на j-ом шаге поиска элемента $C_{ip}$. Если для m файлов условия $\sum_{i=1}^{m} V_i X_{ij} \leq B_j$ выполняются, то алгоритм делает m(n-1) операций сравнения ($E_{ij} \neq 1$) и ($C_{ij} > \max$) на шаге 6 и m операций сравнения $W_j + V_i \leq B_j$ на шаге 7. В таком случае алгоритм делает всего mn операций сравнения. Если условие $W_j + V_i \leq B_j$ не выполняется, то при размещении i-го файла, алгоритм делает p операций сравнения $E_{ij} \neq 1$ на шаге 7б, (n-p) операций сравнения ($E_{ij} \neq 1$) и ($C_{ij} > \max$) на шаге 6 и одну операцию сравнения $W_j + V_i \leq B_j$ на шаге 7, т.е. всего p+n-p+1=n+1 операцию сравнения. Количество невыполненных условий $W_j + V_i \leq B_j$ не больше, чем (n-1). Поэтому временная сложность алгоритма есть

$mn + m(n+1)(n-1) = mn + m(n^2-1) = m(n^2+n-1)$ операций сравнения, а число операций сравнения сделанных алгоритмом: $mn + (n+1)\sum_{i=1}^{m}\sum_{j=1}^{n-1} E_{ij}$ .

Обозначим: P – наилучшее значение целевой функции, полученное полным перебором; A - значение целевой функции, полученное предлагаемым в статье алгоритмом; Q - относительная погрешность алгоритма: $Q = (P-A)\cdot 100\%/P$ . Так как $P \ge A > 0$ , то $0 \le Q < 100\%$. Обозначим: M - максимально возможное значение целевой функции (1), $Q_m$ - максимально возможная относительная погрешность алгоритма: $Q_m = (M-A)\cdot 100\%/M$ . Значение M будет достигнуто, если файл i размещается в узел p, такой, что $C_{i,p} = \max_j C_{i,j}$ , i=1…m, j=1…n. Если A=M, то A=P и Q=0. Если A≠M, то A≤P и $0 \le Q \le Q_m \le 100\%$ .

Если задача имеет большую размерность, полным перебором значение P найти невозможно. Уточним оценку Q для такого случая. Будем считать, что P – значение равномерно распределенной случайной величины $\xi$, которая принимает значения в интервале [A;M]. Наиболее вероятное значение (мода) этой случайной величины совпадает с ее математическим ожиданием. Найдем математическое ожидание $m_\xi$, дисперсию $d_\xi$, среднеквадратичное отклонение $\sigma_\xi$, средневероятное отклонение $\Delta_в$ случайной величины $\xi$ по формулам: $m_\xi = (A+M)/2$ ; $d_\xi = (M-A)^2/12$ ; $\sigma_\xi = \sqrt{d_\xi}$ ; $\Delta_в = (M-A)/4$ . Обозначим: $R = m_\xi + \Delta_в$ ; $Q_r = (R-A)\cdot 100\%/R$ . Таким образом, наиболее вероятно, что $A \le P \le R \le M$ , следовательно $0 \le Q \le Q_r \le Q_m < 100\%$ .

Для исследования точности результатов полученных алгоритмом при решении задачи (1)-(4) проведены три серии экспериментов на ПЭВМ Pentium - 166. Программа составлена на Delphi 3.01, операционная система Windows 98.

В первой серии экспериментов было решено 100 задач методом полного перебора и описанным в разделе 2 алгоритмом. В каждой задаче: m=8, n=3. Элементы матрицы $F_{ij}$ в задачах формировались функцией random(100); $\tau_j$=0,01; $T_j$=100. Вектор V сформирован по формулам: $V_1 = m$ ;$V_i = V_1 - (i-1)$, где i=1…m. $B_1 = \sum_{i=1}^{m} V_i / n$; $B_j = B_1 + (j-1)$, j=1…n. Таким образом, файлы упорядочены по убыванию размеров: $V_i \geq V_{i+1}$, i=1…m-1, узлы упорядочены по возрастанию объема: $B_j \leq B_{j+1}$, j=1…n-1.

Максимальное время работы алгоритма - $T_p = 9 \cdot 10^{-5}$ секунды, время полного перебора - $T_{op} = 5{,}4 \cdot 10^{-2}$ секунды. Время работы алгоритма составляет 0,17 % времени полного перебора. В 27 случаях из ста решения оказались точными. Средняя относительная погрешность алгоритма составила 5,05 %, максимальная - 31,02 %. Для большинства задач относительная погрешность алгоритма не превышает 15 %.

Результаты первой серии экспериментов представлены на рисунке 2.

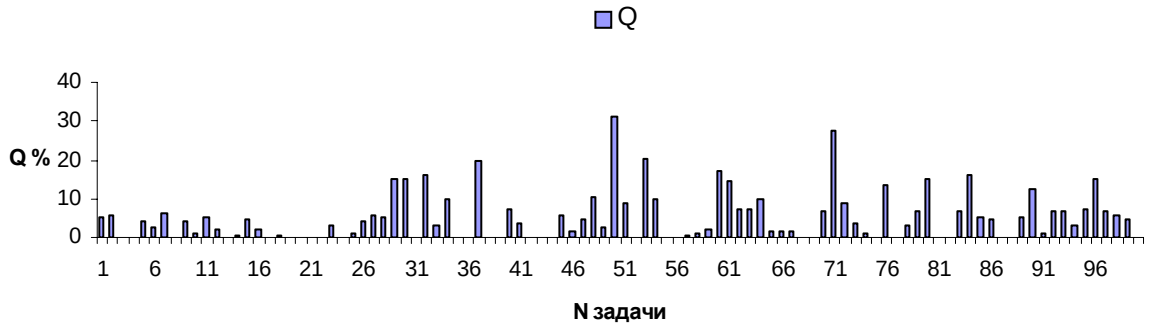


Рисунок 2 - Относительная погрешность решения задач

Во второй серии экспериментов было решено 10 задач для проверки ограничения $A \leq P \leq R$. Условия задач аналогичны условиям первой серии экспериментов. Результаты второй серии экспериментов представлены на рисунке 3 и в таблице 1. Теоретическая относительная погрешность алгоритма превышает реальную относительную погрешность не больше, чем на 10,7 %.

Таблица 1 – Результаты экспериментов

| № | A | M | P | R | $Q$ % | $Q_R$ % |
|---|---|---|---|---|---|---|
| 1 | 224,53 | 235,74 | 224,53 | 232,94 | 0,00 | 3,61 |
| 2 | 210,04 | 220,26 | 210,04 | 217,71 | 0,00 | 3,52 |
| 3 | 203,86 | 206,6 | 203,86 | 205,92 | 0,00 | 1,00 |
| 4 | 173,33 | 196,85 | 185,09 | 190,97 | 6,52 | 9,24 |
| 5 | 207,68 | 215,98 | 207,68 | 213,91 | 0,00 | 2,91 |
| 6 | 212,69 | 246,91 | 225,07 | 238,36 | 5,50 | 10,77 |
| 7 | 238,13 | 250,79 | 238,13 | 247,63 | 0,00 | 3,83 |
| 8 | 158 | 209,57 | 173,57 | 196,68 | 8,97 | 19,67 |
| 9 | 173,45 | 204,11 | 176,12 | 196,45 | 1,52 | 11,71 |
| 10 | 209,41 | 229,1 | 212,44 | 224,18 | 1,43 | 6,59 |

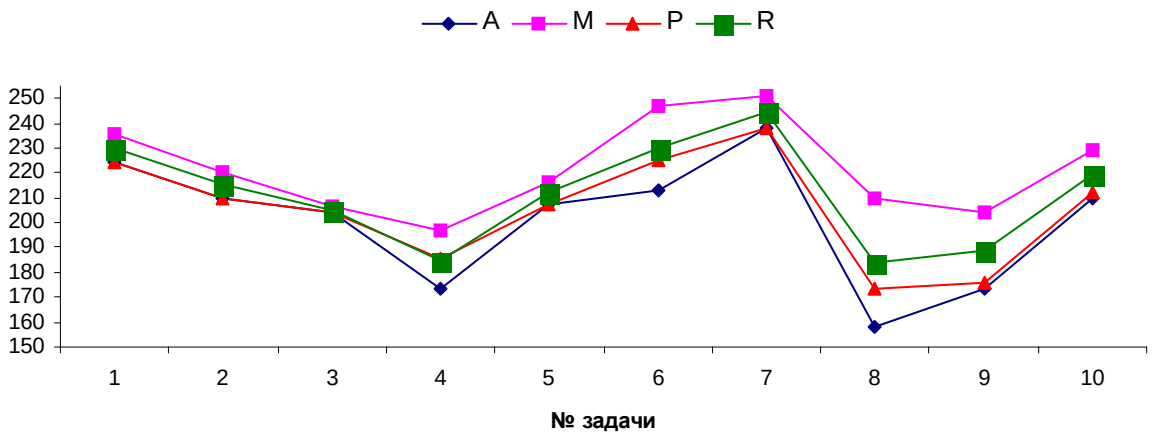


Рисунок 3 - Значения целевой функции в решенных задачах

В третьей серии экспериментов было решено 5 задач большой размерности: m=3000; n=10. Максимальное время работы алгоритма $T_p = 6 \cdot 10^{-2}$ секунды. Максимальная относительная погрешность алгоритма не превышает 2 %. Результаты третьей серии экспериментов представлены в таблице 2.

Таблица 2 – Результаты третьей серии экспериментов

| № | M | A | R | $Q_R$ % | $Q_m$ % |
|---|---|---|---|---|---|
| 1 | 905,06 | 895,72 | 902,72 | 0,78 | 1,03 |
| 2 | 902,59 | 893,86 | 900,41 | 0,73 | 0,97 |

Таблица 2 – Результаты третьей серии экспериментов (окончание)

| № | M | A | R | $Q_R$ % | $Q_m$ % |
|---|---|---|---|---|---|
| 3 | 908,86 | 899,55 | 906,54 | 0,77 | 1,02 |
| 4 | 903,59 | 896,45 | 901,8 | 0,59 | 0,79 |
| 5 | 905,31 | 900,09 | 904 | 0,43 | 0,58 |

В статье предложен метод динамического размещения файлов по узлам вычислительной сети. На его первом этапе определяется интервал между перераспределениями файлов. На втором этапе для перераспределения файлов по узлам вычислительной сети решается задача линейного программирования с булевыми переменными. В качестве оптимизируемого критерия размещения файлов предложена максимизация периода времени между моментами их перераспределения. Ограничениями в задаче служат объемы памяти и коэффициенты загрузки узлов. Для предложенного в статье эвристического алгоритма решения задачи получены оценки временной сложности и относительной погрешности. Временная сложность алгоритма пропорциональна произведению количества файлов на квадрат количества узлов. Относительная погрешность обратно пропорциональна наиболее вероятной величине точного решения. Результаты экспериментов подтверждают полученные оценки и эффективность алгоритма.


1. *Рахлин В.А.* Моделирование машин баз данных распределенной архитектуры. //Программирование, 1996, N 2. - С. 7-16.
2. *Барский В.И.* Вычислительные аспекты решения задачи размещения информационных фондов в сетях ЭВМ. В кн. Вычислительные системы и сети ЭВМ. Москва: 1989. – С. 76-79.
3. *Ладыженский Ю.В., Бельков Д.В.* Рациональное размещение файлов распределенной базы данных в вычислительной сети с произвольной топологией. В кн. Информатика, кибернетика и вычислительная техника (ИКВТ-99). Сборник трудов ДонГТУ, Выпуск 10. Донецк: ДонГТУ, 1999, С. 44-49.