Optimization of distributed file placement registrations on a computer network

Yevhen Davydenko[1,†], Hlib Horban[1,*,†], Alyona Shved[1,†] and Kateryna Antipova[1,†]

[1] Petro Mohyla Black Sea National University St. 68 Desantnykiv 10, 54003, Mykolaiv, Ukraine

Abstract

The article considers a method for optimal placement of data files in a local network, taking into account the criterion of total request service time during the distribution of registrations. This method allows to effectively regulate the server load and use resources with maximum performance, which leads to a reduction in query execution time. An intelligent algorithm for balancing the load of distributed database network nodes that can optimize the processing of large amounts of data is investigated. The research results confirm the possibility of a significant increase in data processing speed through the use of mechanisms for optimizing the load of network nodes.

Keywords

distributed system, computer network, file, request, node, distribution

1. Introduction

The development of information and communication technologies has led to the proliferation of distributed data processing systems based on computer networks. There are scientific and technical tasks of processing ultra-large data sets, the realization of which is not enough for a single computer with a single-processor architecture. An example of an ultra-large database is the EOS/DIS Earth Observation System database, which includes data from many satellites that collect information to study long-term trends in the state of the atmosphere, oceans and earth's surface with a volume of 1 Pbyte of information per year. The Stanford Linear Accelerator Center has a similar system, which has a database created for the BABAR experiment (studying the collision of subatomic particles to determine the impact of matter and antimatter behavior on the formation of the universe) and, has a volume of 1492.0 TB.

Today, real computer networks are characterized by some disadvantages, especially a distributed network is a very heterogeneous medium of information transmission: some sections can be built using ATM or FDDI techniques, while others are based on slow X.25 protocols. The actual speed of information transfer directly depends on the bandwidth of the slowest network. Thus, a remote user's access to a corporate database can be significantly hampered. On the other hand, does a remote user always need absolute access to the entire information base? In many cases, only information that is directly related to their business environment is requested.

The efficiency of an IoT system directly depends on traffic activity, and the lower it is, the faster the funds invested in its construction will pay off [1]. Systematic implementation requires proper

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

davydenko@chmnu.edu.ua (Y. Davydenko); hlib.horban@chmnu.edu.ua (H. Horban); avshved@chmnu.edu.ua (A. Shved); antipova.katerina@chmnu.edu.ua (K. Antipova)

0000-0002-6512-3576 (H. Horban); 0000-0002-0547-3689 (Y. Davydenko); 0000-0003-4372-7472 (A. Shved); 0000-0002-9012-5290 (K. Antipova)

coordination of data distribution and storage. The optimal method for reducing traffic in communication channels is to use the client-server technique [2, 3, 4, 5, 6].

Thanks to the packet switching technique, in-depth and applied research has been conducted in 3 areas. The first direction is related to the development of the basics of packet switching theory in distribution systems [7, 8].

The second area of in-depth research is related to the mathematical theory of optimizing flows in networks and selecting profitable network routes with packet switching [9]. Such research should be conducted, in particular, using methods of expert evaluation [10, 11], the results of which obtained allow to carry out a more profound analysis of the obtained expert information aimed at a synthesis an effective and substantiated group decisions.

The third direction is the implementation of scientific and applied research on the development of modern hardware and software for packet switching technology [7]. In general, research is driven by the need to improve system performance and reliability, reduce overall costs, and expand the range of services provided [7, 8, 9].

Paper [12] investigates the issue of distributing information resources across computer network nodes. Optimization-oriented algorithms for placing information files were considered. The average amount of data sent over communication channels per unit time; total request processing time; total cost of network traffic, etc. were considered as optimization criteria.

There is a need to choose a numerical optimality criterion that determines the average time of user requests execution and is convenient for optimal file placement. The choice of such a characteristic of the mass service system is due to the fact that users are usually interested not in minimizing the size of the queue or any other characteristics of the mass service system, but in ensuring that their requests are processed in as little time as possible.

When determining the average waiting time for requests W in the service queue, it is recommended [13] to use the following formula:

$$W = \frac{\rho^2}{\lambda(1-\rho)},$$

where ρ is the load factor of the service device $(0 \leq \rho < 1)$;

λ is the intensity of the request flow (the average number of packets claiming to be transmitted per unit of time).

When setting the task of optimizing the placement of files among network nodes in order to obtain high quality service, you can keep the average request service time (excluding the service waiting time) constant and independent of the file placement. The value ρ depends on λ and the bandwidth of the serving device μ: $\rho = \lambda/\mu$.

Usually, the maximum allowable waiting time for requests in the service queue M is constant, so the maximum allowable service device load factor is determined from the expression:

$$\rho \frac{M}{M+b_{max}}.$$

When distributing requests among the service devices, they must minimize the value of W, while the route of the request is unknown in advance, i.e., the request can be processed by one service device or by several service devices sequentially.

The target function (quality criterion) is selected as a combination of traffic parameters through communication channels in the network:

$$Q(\rho) = \sum_{i=1}^{N} C_i \rho_i,$$

where C_i is the weighting factors that take into account the average packet service time of the communication channel: $C_i = b_i;\ \ b = \frac{\rho}{\lambda}$.

The problem of creating switching systems designed to analyze the state of the network at any given time and optimize data transportation has not been fully resolved. Systems that perform relatively simple optimization of the distribution of data transmission over network channels remain

extremely expensive, and the efficiency of using the capabilities of universal switches when transmitting large amounts of multimedia information over several channels simultaneously is relatively low.

When providing multi-user access to information resources stored in the form of a database, it is necessary to rationally place the database files in the nodes of a computer network.

There are several relevant mathematical models that differ in the type of objective function and the set of constraints that are taken into account when searching for an optimal method [14].

Today, only application software packages, namely Matlab, are actually used to find optimal solutions [6].

After identifying the optimal solution, model stability and sensitivity analysis is usually performed.

2. Organization of optimization of distributed file placement registrations in a computer network based on the theory of queuing

Let's consider a method of building a model of rational file distribution of a DBMS over the nodes of a computer network, the essence of which is the mathematical apparatus of the concept of queuing. The queue theory is the basis for building a computer network model in many works on optimizing file allocation in a DBMS [9, 15, 16], however, a mass service system with 1 servicing device - a single bus - is taken as a mathematical model of a local network with a bus topology.

Let's analyze the network as a multi-device mass service system, i.e., as a system where several identical service devices process 1 request queue. The request at the very beginning of the queue is sent to one of the free devices for service. The multi-device queue shown in Figure 1 differs from the queue coordination in Figure 2, which shows several single-device queues operating in parallel. If in all cases the service devices and the incoming flow of requests are identical, and in singledevice queues, requests arrive randomly and, once in the queue, remain there (otherwise, moving to another queue is prohibited), then it turns out that the operation of a multi-device queue is preferable to single-device queues operating in parallel.

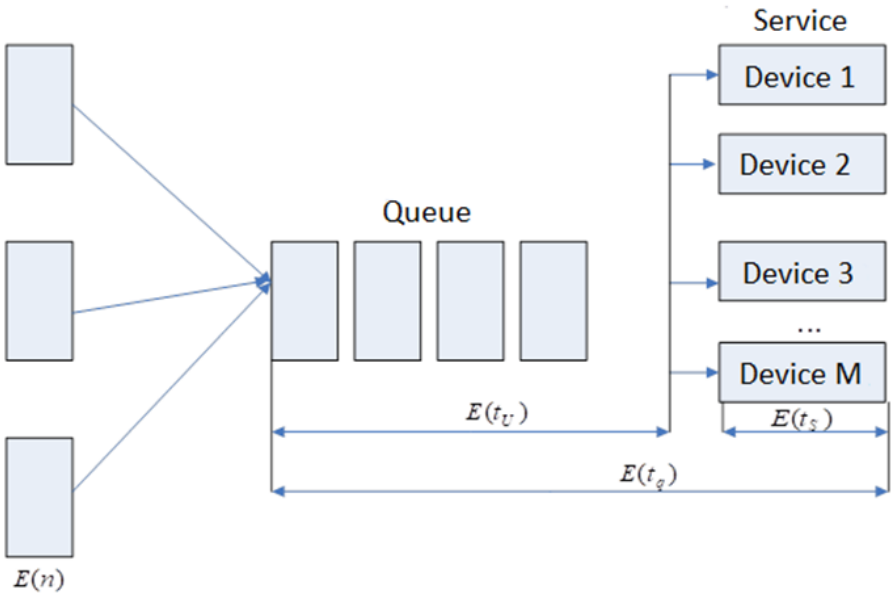

Figure 1: Batch processing when using a queue with a number of service devices.

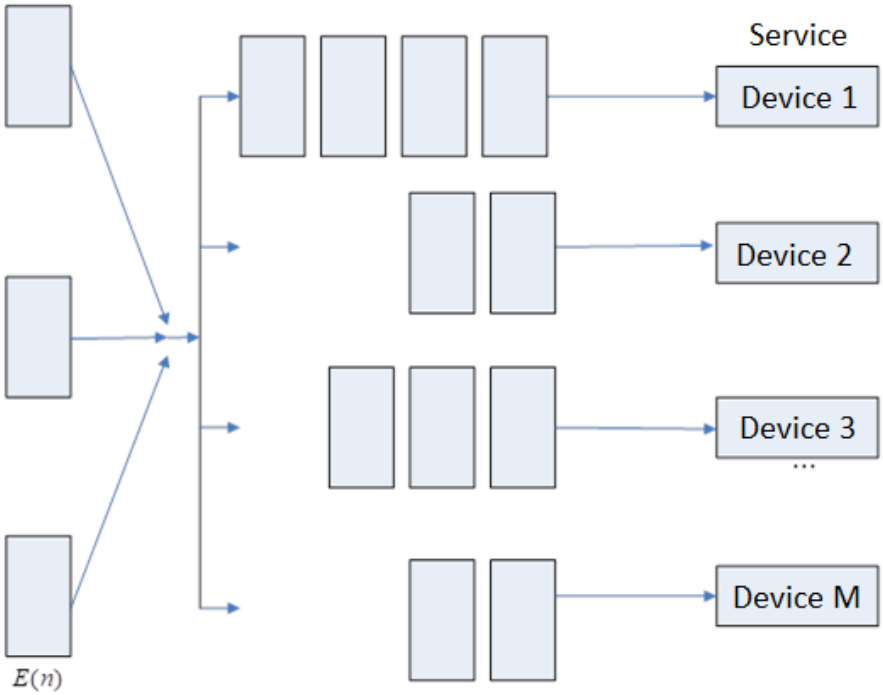

Figure 2: Batch processing when using several parallel queues, with 1 service device.

Let's set the optimal number of copies for each file of distributed databases, considering the computer network as a series of overlapping multi-device SMOs with the 1st queue of requests to a particular file.

During the design of a CMA, worst-case scenarios are often performed. In this situation, the estimates are not very accurate, but at least the errors provide a margin of safety. In real systems, service times fluctuate. The variation can be expressed by calculating the mean and standard deviation for the service time of specific types of equipment. The best case is when the service time is constant, i.e., standard deviation = 0 (i.e., no deviation from the mean). The worst case is also when the maintenance time follows an exponential distribution, i.e., when the standard deviation = the mean (for a standard deviation, this is too high a value, which shows that there is a large spread of maintenance time values). It is worth noting that the exponential distribution is not always the worst case; for example, the mean of 5, 10, 20, and 200 = 58.75, and the standard deviation is approximately 81 [16].

Let's denote the average x by $E(x)$ (mathematical expectation), and the standard deviation x – σ_x. Then:

$$\sigma_x = \sqrt{\frac{(x-E(x))^2}{N}}. \quad (1)$$

Guided by empirical information rather than specific indicators, in the expression $x - \sigma_x$ expression, $N - 1$ is used instead of N, where N is the number of experiments.

Hence:

$$\sigma_x = \sqrt{\frac{(x-\bar{x})^2}{N-1}}, \quad (2)$$

where $\bar{x}$ is the average value of the experimental value.

The estimation of the mean and standard deviation is more accurate the more empirical values are used. When N is larger, the difference between equations (1) and (2) is negligible.

Another method of determining the typical deviation:

$$\sigma_x = \sqrt{E(x^2) - E^2(x)}.$$

Paper [17] shows that 95% of the query response times do not exceed the average response time plus 2 standard deviations. In other words, about 5% of responses take longer than this value.

In order to simplify mathematical calculations, system load is usually expressed in relative terms compared to the maximum load that the system can handle. As a rule, the value is denoted by the letter ρ. As shown in the above definitions, fully utilized equipment has $\rho = 1$ and free equipment has $\rho = 0$. Thus, the equipment utilization rate ranges from zero to one, and is sometimes expressed as a percentage.

There is a rule [18] according to which the response time curve rises sharply when equipment utilization exceeds eighty percent.

When designing a queuing system, the goal is to ensure that its utilization at constant loads is within sixty to seventy percent [9].

The exact method for determining the equipment utilization rate in a mass service system with 1 service device is given by the expression:

$$\rho = E(n)E(t_s),$$

where $E(n)$ is the average number of requests received per unit of service time; $E(t_s)$ is the average time for servicing the 1st request.

Suppose there are M service devices of the same type. So, it sends requests to any device per unit of time $E(n)/M$ requests per unit of time.

Consequently, the utilization rate of a particular device:

$$\rho = \frac{E(n)E(t_s)}{M}. \quad (3)$$

The ratio ρ should be less than one.

Let w be the number of requests waiting to be served at a certain time, and q be the number of system requests waiting and being served at that time.

Let it go on t_w is the service waiting time, and t_q is the time a request stays in the system, i.e. the time it spends both waiting and being served. The average values w, values q, t_w and t_q, let's set it to $E(w)$, q, $E(t_w)$ and $E(t_q)$. Equality is always objective:

$$t_q = t_w + t_s,$$
$$E(t_q) = E(t_w) + E(t_s).$$

Because E(n) is the average number of incoming requests, and then analyzes the steady state

$$E(w) = E(n)E(t_w), \quad (4)$$

and

$$E(q) = E(n)E(t_q).$$

The quantities w, q, t_w and t_q refer to requests waiting on any of the servers M. Substituting in

$$E(q) = E(n)E(t_q) = E(n)E(t_w) + E(n)E(t_s),$$

the corresponding values from equations (3) and (4), we obtain

$$E(q) = E(w) + M\rho.$$

The probability of having N requests in the system at a given time.

$$P(q = N) = \frac{(M\rho)^N}{N!}P_0, \text{ if } N < M,$$

and

$$P(q = N) = \frac{(M\rho)^N}{M!\,M^{N-M}}P_0, \text{ if } N \geq M,$$

where

$$P_0 = \left[\sum_{N=0}^{M-1}\frac{(M\rho)^N}{N!} + \frac{(M\rho)^M}{(1-\rho)M!}\right]^{-1}.$$

The probability that all service devices are busy at a given time is

$$B = P(q \geq M) = \sum_{N=M}^{\infty} P(q = N),$$

and is calculated by the formula

$$B = \frac{1 - \frac{\sum_{N=0}^{M-1} \frac{(M\rho)^N}{N!}}{\sum_{N=0}^{M} \frac{(M\rho)^N}{N!}}}{1 - \rho \frac{\sum_{N=0}^{M-1} \frac{(M\rho)^N}{N!}}{\sum_{N=0}^{M} \frac{(M\rho)^N}{N!}}}.$$

The equation reduces to $B = \rho$ when $M = 1$. The factor B is present in all other equations for systems with multiple service devices. To determine its quantitative indicators, the function B is described, which determines the probability of loading all devices depending on the numerical value of the equipment utilization factor and the number of service devices M.

In a QS with multiple service devices, the average number of requests pending service is

$$E(w) = B\frac{\rho}{1-\rho}.$$

So,

$$E(q) = B\frac{\rho}{1-\rho} + M\rho.$$

A typical deviation for w is:

$$\sigma_w = \frac{1}{1-\rho}\sqrt{B\rho(1+\rho-B\rho)}.$$

Average waiting period before processing is:

$$E(t_w) = \frac{BE(t_s)}{M(1-\rho)}.$$

So, the average time spent in the queue is:

$$E(t_q) = \frac{BE(t_s)}{M(1-\rho)} + E(t_s).$$

Typical waiting period before processing rejection is:

$$\sigma_{t_w} = \frac{E(t_s)}{M(1-\rho)}\sqrt{B(2-B)},$$

and the typical deviation of the time spent in the queue is:

$$\sigma_{t_q} = \frac{E(t_s)}{M(1-\rho)}\sqrt{B(2-B) + M^2(1-\rho)^2}.$$

The probability that the waiting time exceeds t is determined by the following formula:

$$P(t_w \geq t) = Be^{-M(1-\rho)t/E(t_s)}.$$

A real system can be coordinated so that a few requests do not wait for service at all, and a small fraction of them are delayed for a long period of time. In this case, the average waiting time of a delayed request is much higher than $E(t_w)$.

Let's set the average delay time $E(t_w)$ as the average period of time for requests that must wait.

The probability that a request will be in the queue is B. Thus, the average waiting time is

$$E(t_w) = BE(t_d) + (1-B)0 = BE(t_d).$$

But:

$$E(t_w) = \frac{BE(t_s)}{M(1-\rho)}.$$

Therefore:

$$E(t_d) = \frac{E(t_s)}{M(1-\rho)}.$$

The previous equations for queues with a number of servers are based on the assumption that service times follow an exponential distribution. There are no simple expressions that describe multi-instrument QS systems that have better service times than an exponential distribution, but it would be useful to use a mathematical approximation tool to estimate in such situations.

There are several cases where the theory described above is incorrect. The above formulas serve to approximate the most difficult situations that exist in reality. The reason is the assumption of arbitrariness of the request and (sometimes) indicativeness of the service time. In reality, there may be a more favorable request than a random one.

But there are 2 types of situations where queues and delays are much worse than the ones obtained from the above formulas.

First, the maximum number of requests can be received in a short period of time.

In some cases, it cannot be assumed that the arrival time value follows a Poisson distribution. It is worth emphasizing that most forecasting programs for these systems are designed for a Poisson input event stream.

To select a more suitable model from those on offer, it is necessary to assess user requirements. Table 1 shows whether or not the models include the following features of computer networks, information bases, and applications.

Table 1
Comparison of models

	1	2	3	4	5	6
Model I	+	+	-	-	-	-
Model II	+	-	+	+	+	-
Model III	-	+	-	+	+	+

Studying the numerical results of the implementation of models I and II, built with a unified approach, we can emphasize the following features of the models:

1. The obtained examples of tests of the optimal allocation matrix for distributed databases show a huge dependence between the chosen optimality aspect and the final allocation matrix.

2. In the obtained matrices of rational file allocation, when minimizing the average amount of data sent and minimizing the single processing time of absolutely all requests received by the system per unit of time, the assumption of uniform load of network nodes is clearly violated. It is clear that the 1st nodes will be the most loaded.

3. To increase the system throughput, you can apply a restriction on the time it takes to wait for a request from any node as an auxiliary condition. Let a_{ijs} be the waiting time required to execute a request initiated at node K_j to file F_i contained in the s-th node; T_{ij} be the maximum request execution time for file F_i initiated at node K_j. There is a relationship between the values a_{ijs} and T_{ij}:

$$a_{ijs}(1 - x_{ij})x_{is} \leq T_{ij}.$$

For $j \neq s, 1 \leq i \leq m$. In order to obtain constraints from this relation, we need to express the values of a_{ijs} in terms of the variables x_{ij}. This is very difficult to do.

4. The above query processing scheme practically does not fit into the parallelism of information processing in the network, and also does not take into account the very common situation of complex queries (simultaneous access to several files from 1 node). For example, the local database of the host K_j contains the files F_i, and F_{i+1} and the local information database of the node K_{j+1}contains the files $F_{i+1}$and F_{i+2}. The node K_j starts a complex request for the files F_i and F_{i+1}. According to the given scheme, both of these files will be processed in the node K_j in turn. However, it would be more logical to send a request to process file F_{i+1} loaded) when searching for file $F_i$on node K_j.

5. However, if the issue is solved in a comprehensive manner, i.e., by software optimization and even hardware upgrades, then the load of some network nodes will not affect the speed of operation. Therefore, despite the above disadvantages, these models of the optimization problem can be applied in practice when designing certain databases.

Thus, the proposed mathematical models of the optimization problem of file allocation of the DBMS on the nodes of the local network can be successfully applied in the design of certain distribution

databases, using a preliminary assessment of user requirements and a software package for the purpose of statistical collection and optimal redistribution of requests.

3. Coordination of optimization of file placement of the database by a single time of request service

Several works [8, 9, 14, 16] have been devoted to solving the problem of rational placement of information files on local network nodes, which differ in both the problem statement and the methods of its solution.

We study a network with a single bus topology. Local area networks with a bus topology are characterized by relative ease of management, low arbitration time, ease of expansion, and fairly high reliability (due to parallel connections of nodes to the channel) [9].

Let's say that a query that comes to any network node involves access to a database file. We will distinguish between 2 types of requests: search requests and fix requests. Queries are served in the node in the order of receipt. To save resources, we do not implement a priority system. A search request is initiated in a specific node. If a copy of the required file is contained in the local node database from which the request came, it is processed. If a copy of the required file is not in the local database of this node, the search request is sent to a free node that contains a copy of the required file, processed there, and the result is sent to the original node.

As an aspect of rationality, a single time required to service requests received by the system within a unit of time is accepted. The bus topology, the uniformity of communication lines and their short length in local area networks make the sending time independent of the request node and the transmission node.

Let:

- n is the number of network nodes;
- m is the number of independent files of distributed databases;
- K_j is the j-th steam node;
- F_i is the i-th file of distributed databases;
- L_i file size F_i;
- b_j is the storage capacity of the K_j node, which is intended to host files;
- s is the number of search query types;
- λ_{kij} is the intensity of k-type search requests to the file F_i from the node K_j;
- t_{kij} is the processing time of a k-type search request to the file F_i in the node K_j;
- $T_{ki}^{(1)}$ is the time of sending a k-type search request to the file;
- $T_{ki}^{(2)}$ is the time it takes to send a response to a k-type search request to the file F_i;
- r is the number of types of corrections;
- $\lambda'_{lij}$is the intensity of l-type fixes of the file F_i from the node K_j;
- t'_{lij} is the processing time for fixing the l-type of the file F_i in the node K_j;
- $T'_{li}$is the time of sending the l-type file patch F_i;
- $x_{ij}(i = 1, \dots, m; \ = 1, \dots, n)$are the values determined by the formula:
  - $x_{ij} = 1$, if a copy of the file F_i is located in the node K_j,
  - $x_{ij} = 0$, if a copy of the file F_i is not located in the node K_j.

The time it takes to send data from the node K_j data during the execution of a k-type search query to the file F_i, is equal to $\left(T_{ki}^{(1)} + T_{ki}^{(2)}\right)(1 - x_{ij})$. Then the only time required to send data through

communication channels between nodes when executing search queries received by the system during a unit of time is set as

$$T = \sum_{k=1}^{s}\sum_{i=1}^{m}\sum_{j=1}^{n}\lambda_{kij}\left(T_{ki}^{(1)} + T_{ki}^{(2)}\right)\left(1 - x_{ij}\right).$$

By accepting

$$a_{ij} = \sum_{k=1}^{s}\lambda_{kij}\left(T_{ki}^{(1)} + T_{ki}^{(2)}\right),$$

we get

$$T = T_0 - \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}x_{ij},$$

where $T_0 \equiv \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}$.

The time it takes to send from the node K_j information when performing a l-type file correction F_i is equal to $\sum_{p=1,p\neq j}^{n} T'_{li}x_{ip}$.Then the only time required to send information over communication channels between nodes when fulfilling adjustment requests received by the network within a unit of time is set as

$$T' = \sum_{l=1}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{p=1,p\neq j}^{n}\lambda'_{lij}T'_{li}x_{ip}.$$

If you put

$$a'_{ij} = \sum_{l=1}^{r}\lambda'_{lij}T'_{li},$$

then

$$T = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}\sum_{p=1,p\neq j}^{n} x_{ip} = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}\sum_{p=1,p\neq j}^{n} a_{ip} = \sum_{i=1}^{m}\sum_{j=1}^{n}\hat{a}_{ij}x_{ij}\,,$$

where $\hat{a}_{ij} = \sum_{p=1,p\neq j}^{n} a_{ip}$

Processing time for a k-type search request to a file F_i from the node K_j can be represented by the formula

$$t_{kij}x_{ij} + \hat{t}_{kij}\left(1 - x_{ij}\right) = \hat{t}_{kij} + \left(t_{kij} - \hat{t}_{kij}\right)x_{ij},$$

where $t_{kij} = \frac{1}{n-1}\sum_{p=1,p\neq j}^{n} t_{kip}$.

The only time required to process all search queries received by the network during a unit of time is set as

$$t = t_0 + \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}x_{ij}\,,$$

where

$$t_0 = \sum_{k=1}^{s}\sum_{i=1}^{m}\sum_{j=1}^{n}\lambda_{kij}\,\hat{t}_{kij},$$

$$b_{ij} = \sum_{k=1}^{s}\lambda_{kij}\left(t_{kij} - \hat{t}_{kij}\right).$$

The time it takes to process a fix for an l-type file F_i is equal to

$$\sum_{q=1}^{n} t'_{liq}x_{iq}.$$

The standardized time required to fulfill all patch requests that come into the network during a unit of time is set as

$$t' = \sum_{l=1}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{q=1}^{n}\lambda'_{lij}t'_{liq}x_{iq} = \sum_{l=1}^{r}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{q=1}^{n}\lambda'_{liq}t'_{lij}x_{ij}.$$

By putting

$$b'_{ij} = \sum_{l=1}^{r}\sum_{q=1}^{n}\lambda'_{liq}t'_{lij},$$

we get

$$t' = \sum_{i=1}^{m}\sum_{j=1}^{n}b'_{ij}x_{ij}.$$

Noting $f_0 \equiv T_0 + t_0, C_{ij} = -a_{ij} + \hat{a}_{ij} + b_{ij} + b'_{ij}$,we obtain an exact model of the problem of optimal distribution of copies of files between network nodes in terms of the minimum uniform time required to service all requests received by the system within a unit of time, associated with the type of discrete programming problems with boolean variables:

$$f(X) = f_0 + \sum_{i=1}^{m}\sum_{j=1}^{n}C_{ij}x_{ij} \to min, \quad (5)$$

under restrictions

$$\sum_{j=1}^{n}x_{ij} \geq 1 \ (i = 1,2,\dots,m); \quad (6)$$

$$\sum_{j=1}^{n}L_i\,x_{ij} \leq b_j (j = 1,2,\dots,n); \quad (7)$$

$$x_{ij} \in \{0,\ 1\}(i = 1,2,\dots,m;\ \ j = 1,2,\dots,n) \quad (8)$$

4. Algorithmic implementation of the model

To implement models (5)–(8), we propose an algorithm that creates a model [14] for further comparison.

At the first stage of the algorithm, the initial distribution of files is found, which will be rational if the condition (7) is not taken into account. At the second stage, the files are redistributed if there is at least a 1-n index for the original distribution.

$\mathcal{J}$ is such that condition (7) is not met. The second stage of the algorithm is performed until a distribution is found that meets condition (7). Let us consider the stages of the recommended algorithm.

The first stage. Determination of the initial distribution.

Determining the values of $C_{ij}(i = 1,2,\dots,m;\ \ j = 1,2,\dots,n)$ and calculation of the matrix $C = C_{ij}{}_{mn}$.

If for $i\ \exists\ C_{ij} < 0$,then

$$x^*_{ij} = \begin{cases}1, C_{ij} < 0;\\ 0, C_{ij} \geq 0.\end{cases}$$

If for $\forall C_{ij} \geq 0$, then we define $min_{1\leq j\leq n}C_{ij}$. Let $min_{1\leq j\leq n}C_{ij} = C_{ij_i}$.

Then:

$$x^*_{ij} = \begin{cases}1, j = j_i;\\ 0, j \neq j_i.\end{cases}$$

The second stage. Redistribution of files.

1. Create a vector of values $E = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$, where $\varepsilon_j = 0\ (j = 1,2,\dots,n)$. During the algorithm, after redistributing a file from a certain filled node K_j the corresponding component ε_j of the vector E is set to 1, and this node is closed for redistribution.

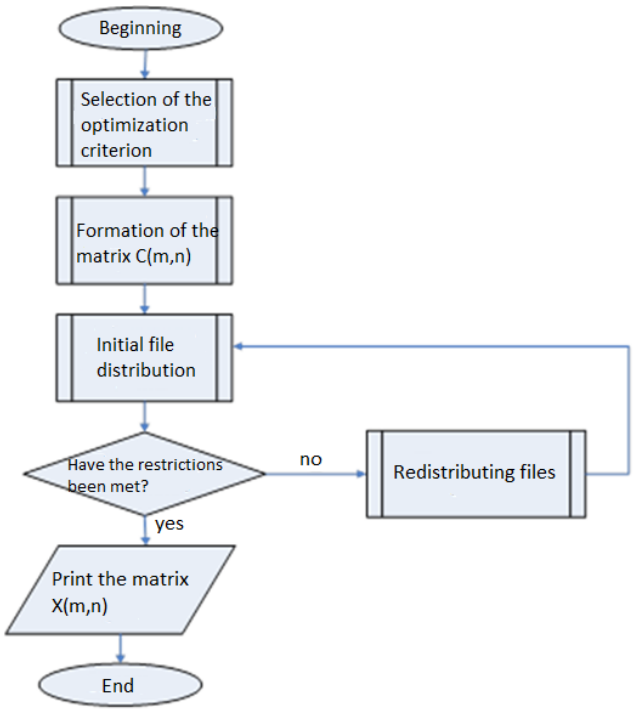

Figure 3: Main algorithm.

2. For all indices j, where $\varepsilon_j = 0$, we check the fulfillment of condition (8). If this condition is fulfilled for all j indexes, then the algorithm ends. If for sure j=r we have:

$$\sum_{i=1}^{m} L_i x_{ir} > b_r,$$

Then we move on to the third point.

3. If $\exists\, s \neq r$ such as:

$$\sum_{i=1}^{m} L_i x_{ir} \leq b_s, \sum_{i=1}^{m} L_i x_{is} \leq b_r,$$

then we swap the memory of the r-th and s-th nodes and return to the second point. Otherwise, we go to the fourth point.

For those where $\exists j \neq r$ such as $x_{ij} = 1$, visualize $min_i(-C_{ir})$. Let:

$$min_i(-C_{ir}) = -C_{kr}.$$

For those i, where $x_{ij} = \begin{cases} 1, & j = r, \\ 0, & j \neq r, \end{cases}$ we determine $min_j(C_{ij} - C_{ir})$, where little is taken, according to those indicators $j \neq$, where $\varepsilon_j = 0$. Let:

$$min_j(C_{ij} - C_{ir}) = C_{ij_i} - C_{ir}.$$

Then we define $min_j(C_{ij} - C_{ir})$. Let:

$$min_i(C_{ij_i} - C_{ir}) = C_{lj_l} - C_{lr}.$$

If $min(-C_{kr}, C_{lj_l} - C_{lr}) = -C_{kr}$, then in the matrix X we assume $x_{kr} = 0$. This means that the file F_k is excluded from the node K_r. If $min(-C_{kr}, C_{lj_l} - C_{lr}) = -C_{lr}$, then in the matrix X provide $x_{lr} = 0$, $x_{lj_l} = 1$.This means that the file F_i from node K_r is redistributed to node K_j. Such a redistribution of files corresponds to a minimal increase in the objective function.

Check condition (7) $j = r$. If it is not fulfilled, then go to the third step. If the condition is met, then the element ε_r element of the vector E is assigned a value of 1 and proceed to the second step.

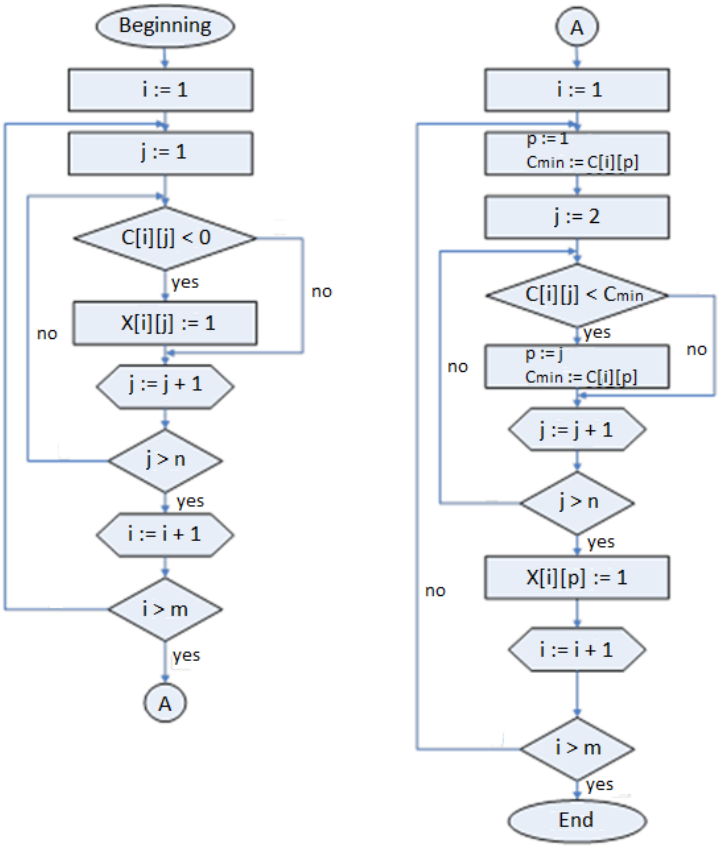

Figure 4: File distribution.

Thus, the algorithm allows you to find the optimal or almost rational distribution of files between network nodes in a finite number of steps. The result of the algorithm is the matrix X.

5. Optimization of the distribution of registrations

To determine the intensity of access to various files of the information base, we used the resident program "Query Analyzer" written in the Assembler programming language. The language guarantees compactness and flexibility when writing resident programs and does not allow errors in the measured processes. The resident program is loaded on all network nodes. The computer time is synchronized.

The program analyzes all file requests and records the date, time, file name, request duration, response time, and response duration.

Logging is performed in the internal buffer and is written to disk only during computer idle time, when typing from the keyboard or other operations that require waiting for a response are performed.

The non-resident part of the program analyzes the processed data and finds temporary file access properties. The network load is characterized by unevenness - complete absence of calls or simultaneous calls from all workstations.

The LAN Query Optimizer program is designed to redistribute search queries for reference and information files between network nodes, which significantly reduces queues.

As an aspect of optimality, a single time required to service all requests received by the system in 1 hour is taken as a single time.

The efficiency of requests for corrections and searches in various data files of the Revenue Accounting DBMS and the average search time are presented in the table.

The output file for the optimization program was created in the following order: the efficiency of searches and corrections in any database file; the duration of records in the files; and the average processing time of search queries to data files.

As an aspect of optimality, the average amount of information sent over the communication lines when processing requests is taken. The resulting matrix of the expedient distribution of files to network nodes as a result of the calculation using the optimization program is as follows:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

For example, files that have been modified are often stored in 1 copy each. Copies of other files are duplicated in the local databases of network nodes.

If we take a single time as the optimality aspect, the time required to service all requests that come into the system within 1 hour, the matrix of the appropriate file allocation is of a different type:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For example, the first 4 files of information bases, which have a high intensity of corrections, are stored on the 1st of the most productive nodes (1 copy each).

Copies of other files, where corrections and additions are made less frequently, are located in the local databases of the first four computer nodes.

In fact, it is more convenient to use the second method of file distribution, since for very time-limited work on the introduction of payments, distribution of incoming amounts between budgets and the creation of general reporting, it is enough to use the first four network nodes with the highest speed. Similar placement of copies of data files will help to avoid unnecessary information redundancy and difficulties in reconciling it.

6. Conclusion

The method of optimization of distributed file placement registrations in a computer network based on the waiting time. The problem of creating switching systems designed to analyze the state of the network at any given time and optimal data transportation to find optimal solutions.

A practical implementation of the method of balancing the load of the DBMS network nodes intended for processing large and ultra-large volumes of databases is proposed. The results of the operation of the revenue accounting system based on the proposed method indicate the possibility of a significant increase in the speed of data processing in large-volume databases by using mechanisms for optimizing the load of network nodes used to process databases. The application of the method allows to increase the productivity and reduce the reaction time of the information system working with the DBMS.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Krainyk Y., Davydenko Y., Tomas V. Configurable Control Node for Wireless Sensor Network. 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT), Lviv, Ukraine, 2019, pp. 258–262. doi: 10.1109/AIACT.2019.8847732

[2] Bailis P., Ghodsi A., Braams J., Hellerstein H., Stoica I. Bolt-on causal consistency. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013. pp. 761–772.

[3] Boncz P., Zukowski M., Nes N. LATEX: MonetDB/X100: Hyper-Pipelining Query Execution. Cidr, Vol. 5, 2005. pp. 225–237.

[4] Charapko A., Ailijiang A., Demirbas M. Adapting to Access Locality via Live Data Migration in Globally Distributed Datastores. In 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018. pp. 3321–3330.

[5] Liu G., Shen H. Minimum-cost cloud storage service across multiple cloud providers. IEEE/ACM Transactions on Networking (TON), Vol. 25, 4 (2017). pp. 2498–2513.

[6] Mahmoud H., Nawab F., Pucher A., Agrawal D., El Abbadi A. Low-latency multi-datacenter databases using replicated commit. Proceedings of the VLDB Endowment, Vol. 6, 9 (2013), pp. 661–672.

[7] Guerraoui R., Wang J. How fast can a distributed transaction commit? In Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. ACM, 2017. pp. 107–122.

[8] Ping F., Hwang J.-F., McConnel C., Vabbalareddy R. Wide area placement of data replicas for fast and highly available data access. In Proceedings of the fourth international workshop on Data-intensive distributed computing. ACM, 2011. pp. 1–8.

[9] Ports D., Grittner K. Serializable snapshot isolation in PostgreSQL. Proceedings of the VLDB Endowment, Vol. 5, 12 (2012), pp. 1850–1861.

[10] Shved A., Kovalenko I., Davydenko Y. Method of Detection the Consistent Subgroups of Expert Assessments in a Group Based on Measures of Dissimilarity in Evidence Theory. In: Shakhovska N., Medykovskyy M. (eds) Advances in Intelligent Systems and Computing IV. CCSIT 2019. Advances in Intelligent Systems and Computing, vol 1080. Springer, Cham, 2020. pp. 36–53. doi: 10.1007/978-3-030-33695-0_4

[11] Kovalenko I., Davydenko Y., Shved A. Formation of Consistent Groups of Expert Evidences Based on Dissimilarity Measures in Evidence Theory. 2019 IEEE 14th International Conference

on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2019, pp. 113–116. doi: 10.1109/STC-CSIT.2019.8929858
[12] Adya A., Myers D., Howell J., Elson J., Meek C., Khemani V., Fulger S., Gu P., Bhuvanagiri L, Hunter J., et al. Slicer: Auto-sharding for datacenter applications. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016. pp. 739–753.
[13] Lakshman A., Malik P. Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, Vol. 44, 2 (2010), pp. 35–40.
[14] Bacon D., Bales N., Bruno N., Cooper B., Dickinson A., Fikes A., Fraser C., Gubarev A., Joshi M, Kogan E., et al. Spanner: Becoming a SQL system. In Proceedings of the 2017 ACM International Conference on Management of Data. 2017. pp. 331–343.
[15] Klophaus R. Riak core: Building distributed applications without shared state. In ACM SIGPLAN Commercial Users of Functional Programming. ACM, 2010. p. 14.
[16] Pavlo A., Angulo G, Arulraj J., Arulraj H., Lin H., Lin H., Ma L., Menon P., Mowry T., Perron M., Quah I., et al. Self-Driving Database Management Systems. In CIDR, Vol. 4. 1 (2017).
[17] Saha D., Ghatore G. S., O'Brien B. DAT202: Getting started with Amazon Aurora. 2017. URL : https://www.slideshare.net/AmazonWebServices/dat202getting-started-with-amazon-aurora/14 (Last accessed: 11.01.24).
[18] Nishtala R., Fugal H., Grimm S., Kwiatkowski M., Lee H., Li H. C., McElroy R., Paleczny M., Peek D., Saab P., et al. Scaling memcache at facebook. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). 2013. pp. 385–398.S. Anzaroot, A. McCallum, UMass citation field extraction dataset, 2013. URL : http://www.iesl.cs.umass.edu/data/data-umasscitationfield.