УДК 004.3

THE PRINCIPLES OF THE ORGANIZATION OF THE GRID COMPUTING IN COMPUTER NETWORKS

Solya V.P.
Educational-Scientific Complex «Institute for Applied System Analysis»
of the National Technical University of Ukraine «Kyiv Polytechnic Institute»

Principles of the organization of grid computing in corporate computer networks are investigated in article. The problem of development of new management and control methods of the grid calculations in the enterprise networks with the purpose of minimization of problems decision time due to improvement of the separate task distribution mechanism inside of a network is solved.
Keywords: grid, coordinator routine, execution manager, agent task, aggressiveness, reliability.

Currently, there is a mismatch in the growth of computer processing power, on the one hand, and the spread of networks and their capacity, on the other.

This permits separate independent researchers to use distributed computing systems based on common local and corporate computer networks. Organization of computations in such networks has some fundamental differences from similar problems for wide-area networks (**GRID** systems). The differences lie primarily in the following points:

· compilation of an optimal match of individual computing fragments to accessible computers (computing elements);

· improvement of the reliability of computations;

· control of the load on individual computers.

Principles of organization of distributed computing for such networks have not been developed sufficiently as of the present.

The presence of these factors makes the task of developing new charts of organization and management of distributed computing that are tailored to these computer networks particularly urgent.

The object of research is the environment of organization of distributed computing in computer networks.

Computer networks that are considered in this case have a number of features:

· corporate and local area networks are composed of a relatively small number of computers (up to several hundred) that are connected into local networks;

· local networks are often removed from each other geographically and are connected via the Internet;

· computers that are included into such a network may substantially differ in terms of hardware specifications, physical accessibility, type and reliability of network connection, and usage modes;

· computers are inalienable, i.e. they are not passed on for solution of computation-intense operations completely but they can run user tasks with a higher priority.

The goal in this case is to minimize the time required to solve computation-intense operations by improving the mechanism for distribution of individual fragments of (sub)tasks within a computer network.

To achieve this goal, it is necessary to solve the following tasks:

· study of peculiarities of computers and communications in target networks;

· analysis of the features of organization of distributed computing in such networks;

· development of a mathematical model of the process of distributed computing, which takes into account the features of the runtime environment for computations and solved tasks;

· development and implementation of a simulation model of a process of distributed computing in the form of a program;

· synthesis of an algorithm for distribution of tasks with the use of a mathematical model;

· study of the effectiveness of the synthesized algorithms with the use of software implementation of a simulation model;

· validation of algorithms with the use of practical tasks.

The algorithm is focused on large-volume computing tasks, which are broken down into subtasks, with the minimum computation time starting at an hour or more of computer time. For tasks that can be divided into less demanding subtasks (owing to the increase in their quantity), the issues of the optimal distribution usually do not emerge because of negligible loss of time at the loss of results of solutions because of failure of a computer with exception, naturally, of a computer that stores the results of computations for all the subtasks of the task to be solved at the time.

Description of the Distributed Computing Organization System

The core of the system under study is the client-server architecture [5], which is characterized by the presence of a server computer that is kept in the idle state while awaiting queries from client computers. Upon receipt of such a query, the server processes it and sends the result to the client.

The model of interaction of processes in the system is as follows: a control station – workstations [5]. This system diagram features a special station (a control station) that has a set of subtasks (a portfolio). The control station distributes subtasks among computing elements (workstations) and collects the results. This is a variant of the client-server diagram in which workstations act as computing servers and the control station is the client.

As a software solution, the system includes the following components:

· **coordinator routine:** a program that runs on the server. This program is responsible for regis-

tration of computing elements and their allocation to address individual tasks in accordance with the existing policy of the coordinator routine;

· **execution manager:** a program that runs on any computer within the network. The execution manager corresponds to a single task being solved and is responsible for organizing the process of computation by: creating subtasks, sending subtasks to the computer station, and saving and analyzing the results of computations; stations are dynamically allocated to the execution manager by the coordinator;

· **agent task:** a program that runs on computing elements. It performs three tasks:

– collection of statistics on the functioning of the computing element;

– registration of connection of a computing element to the network with the coordinator routine;

– receipt of subtask data from the execution manager, implementation of actual computations that are needed to solve subtasks, and sending the results back to the execution manager.

Any task that is solved in the system must support partitioning into subtasks, which consist of a software module, which runs the required computational algorithm, and input data for the algorithm. Subtasks are numerically characterized by resource intensity, which refers to the amount of computation in certain conventional elementary operations to be performed by a computing element for its solution. More details on this parameter are provided below.

This work studies aspects of interaction between the execution manager and computing elements. Accordingly, the selection of a new CE by the coordinator routine for the solution of the current task can be viewed as connection of the CE; and removing the CE from the task, as a failure.

Let us consider the process of generation and distribution of subtasks in more detail.

At any given time, a queue of computational subtasks and a set of available processing elements exist.

Subtasks are divided into three categories according to the type of event that caused this subtask to be generated:

· starting (initiation of the computation process);

· generated (completion of a computation subtask);

· repeated (failure of a computing element).

Starting subtasks are subtasks that are queued at the time of the initial start of the computation, i.e. they correspond to the initial breaking of the task.

Generated subtasks are based on the analysis of already completed subtasks (for example, in case of insufficient accuracy of the obtained results or in case of non-compliance with conditions for computation completion).

Repeated subtasks are subtasks that have already been transmitted to computing elements but have not been completed for some reason. Reasons for cancelation may be either explicit (with a message on the termination of the computation as a result of software failure) or indirect (with loss of communication with an element for a long period of time).

In general, it is impossible to predict the number of subtasks that are simultaneously queued as a result of occurrence of an event of the first or the second type. Ideally, the flow of repeated subtasks has a zero intensity.

The stream of generated subtasks is associated with the flow of serviced subtasks with a certain dependence, which is determined by the nature of the computational algorithm, and can also have a zero intensity. At the same time, it is necessary to take into account that generated subtasks can be more demanding than previous ones due to, for example, the need to improve the accuracy of computations for a sustainable solution. In other words, the time required for the solution of generated subtasks by the same workstation may increase progressively as the subtask is solved.

Absence of subtasks from the queue means completion of the computation process provided that all the subtasks that have been assigned to elements have been completed.

A subtask can be deleted from the queue as a result of one of two events:

· assignment of the subtask to a computing element (transmission to service);

· abortive termination of computations.

All the currently available computing elements form service equipment (SE) or service channels. The number of elements may vary as a result of the following events:

Quantity expansion:

· a new element has been connected to the computer network (this case includes elements that were occupied by user tasks);

· the computing element has finished processing the assigned subtask (either the subtask has been completed or a processing failure has occurred) and is ready to accept a new subtask;

Quantity reduction:

· assignment of a task to the computing element;

· occurrence of the period of unavailability of the computing element (disconnection of the element from the network or receipt of a resource-intensive user subtask).

That is, SE can be in one of three states:

· free;

· application processing;

· failure.

The general system diagram is shown in Figure 1.

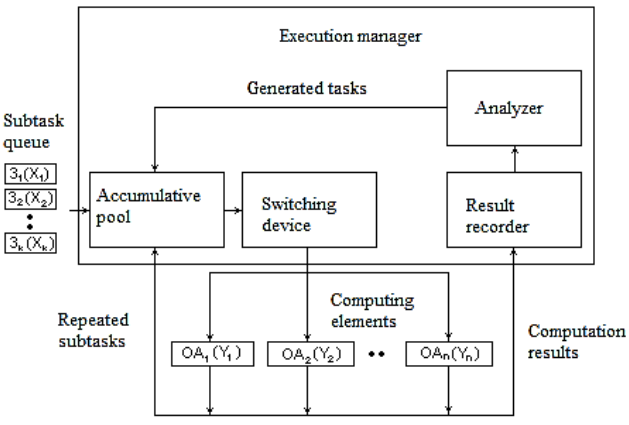

Fig. 1. System Diagram

It is necessary to create assignments of subtasks to computing elements so as to minimize the estimated computation time.

To solve this task, we suggest introducing the following characteristics of computing elements:

· computational power;
· aggressiveness;
· reliability.

Computational power is a numerical assessment of performance of a computing element, which is calculated as the inverse ratio of the time required to perform a set of computing tests to the time that is required to perform the given set of tests with the use of a reference machine.

Aggressiveness is a characteristic that depends on the time and that is based on statistical data that are collected in the computing element. This indicator starts to increase before the time segments wherein the resources of the computing element will not be used by the owner for a long time (for example, the computer is left on while the owner is at work) and falls to zero before the time segments wherein the computing resources are completely occupied or are not available at all (the computer is turned off at night). Moreover, the aggressiveness value is integral with respect to the amount of free resources in the subsequent period of time, i.e. it will be the maximum before the long period of inactivity.

Because aggressiveness begins to increase before the actual release of resources occurs, the coordinator will be able to implement a more «visionary» distribution of tasks without having detailed statistical data about specific elements but only in possession of overall estimates.

Mathematically, aggressiveness is defined as follows:

$$a = \frac{\int_{t_1}^{t_2} (1 - L(t))dt}{t_2 - t_1},$$

where:

a is the numerical value of the aggressiveness;

t_1 is the planned subtask solution start time;

t_2 is the estimated subtask solution end time;

L(t) is the load of the user with tasks at a moment in time t.

The t_2 value is calculated on the basis of the anticipated resource intensity of the subtask, the load of the computing element, and its power:

$$\int_{t_1}^{t_2} P \cdot (1 - L(t))dt = C,$$

where:

P is the numerical value of the power of the computing element;

C is the computing resource intensity of the subtask.

The t_2 value is determined iteratively by increments with the use of numerical integration with a step equal to the statistics locking interval until the integral value exceeds the resource intensity of the subtask.

Evaluation of resource intensity of a subtask is performed by the developer as subtasks are generated on the basis of test runs. Estimation is made in computational power units on the basis of comparison of the runtime to the reference task.

If the (t_1; t_2) interval features statistical probability of failure of the element that exceeds a certain threshold, the aggressiveness will be considered to equal zero.

Reliability is inversely proportional to the number of failures of received subtasks through the fault of the computing element for a certain period of time.

A set of statistical data collected by the agent task on a computing element includes:

· the average execution time of one subtask given its resource intensity;
· the factor of load of computers with user tasks (from 0, which indicates the absence of loads, to 1, which indicates full loading with user tasks);
· the number of failures in a certain time interval;
· periods of unavailability of the element;

Statistics are collected at such a time interval that reveals the frequency of changes in statistical values (for example, daily, weekly) depending on the mode of use of the computer.

The use of statistical data creates additional problems to be solved in the framework of operation:

· development of an algorithm to determine the duration of the statistics collection interval;
· development of effective methods of computation of aggressiveness in case of discrepancy between the real-time mode of use of computers at the current time interval and the predicted mode of use based on statistics [6].

Subtask Allocation Algorithm

In order to solve the subtask allocation task, we recommend to use the mathematical apparatus of the queuing theory [1, 2].

Building a mathematical model is based on the following assumptions:

· for each CE that is occupied with subtask computation, the probability of release at a certain time interval can be calculated;
· for any CE, the probability of failure/inaccessibility at any time interval can be determined;
· for each subtask, the estimated solution time on any CE at any interval of time can be calculated (with the use of statistical data or methods of predictive modeling of the load);
· with the knowledge of the nature of computation (iterative computation, computation with an increasing number of subtasks etc.), the probability of a certain number of subtasks of a specific resource intensity at the next time interval can be predicted;
· upon analysis of the data on the flow of subtasks (change of their number and resource intensity) and the likelihood of completion of computations with the use of CE, the resource intensity of subtasks that may be generated at the next time interval can be predicted.

The methods of identification and analysis of relevant statistics are provided in [3, 4, 7].

The algorithm is based on the idea of partitioning computing elements into groups according to their aggressiveness and on similar classification of received resource-intensity subtasks so that each group corresponds to own subgroup of computing elements. When assigning a subtask, a computing element that is part of the group of not less than the task class is selected.

Thus, we are guaranteed to minimize the execution time required for the calculation of the most resource-intensive subtasks, which reduces the consequences of occurrence of possible failures. Partitioning into groups simplifies the solution while al-

lowing for handling the concepts of group resource intensity and its volume (for subtasks) and handling the probability of completion of computations by means of a computing element of a particular group, transfer thereof to another group, or probability of failure (for groups of computing elements). It is also possible to establish the relationship between resource intensity of the already-solved subtask and the number and resource intensity of subtasks that were generated on the basis of the results of its computation by analyzing the set of solved subtasks of a certain group.

The core of the idea of partitioning is the estimation of the possible time for calculation of subtasks with the use of computing elements of a certain group and the likelihood of occurrence of more resource-intensive subtasks in the computation interval.

Analysis of subtasks can also allow for classification of the problems being solved into the following types on the basis of a number of features:

In terms of the nature of computations [5]:

· Iterative computations;

· Recursive computations.

In terms of the resource intensity of generated subtasks in the course of solution of a single subtask:

· Decreasing resource intensity;

· Constant resource intensity;

· Increasing resource intensity.

In terms of the number of generated subtasks per subtask of a specified type:

· Without generation;

· One to one;

· One to many.

This classification will highlight the most effective tactics of distribution of subtasks for different combinations of types and select the sets of parameters that can be taken into account when partitioning tasks into subtasks. Thus, for iterative algorithms, which require completion of computations for all the subtasks in the current step for the implementation of the next step, it is also possible to effectively balance the resource intensity of generated subtasks while analyzing the parameters of groups of computing elements.

A **simulation model** of distributed computing that is required to study the synthesized algorithm has been implemented as a program that performs step-by-step modeling of the task resolution process in a distributed environment, which is described by means of statistical data.

The model takes the following factors into account:

· inconstancy of composition of a computer network;

· variation of computational power of individual elements;

· availability of random failures during computation as well as in the course of transmission of the tasks and the results;

· different resource intensity of subtasks transmitted for processing

· inaccuracy in the preliminary assessment of the resource intensity of subtasks;

· accounting of the load of CE with user tasks.

Implementation of the method of molecular dynamics in reference to physical chemistry problems [8, 9] as an applied task is planned.

References:

1. Saul'ev V. K. Matematicheskie modeli teorii massovogo obsluzhivanija [Mathematical models of queuing theory] / V. K. Saul'ev. – M.: Statistika, 1979. – 96 p. (in Russian).
2. Hinchin A. J. Raboty po matematicheskoj teorii massovogo obsluzhivanija [Works about the mathematical queuing theory] / A. J. Hinchin. – M.: Fizmatgiz, 1963. – 236 p. (in Russian).
3. Kruglikov V. K. Verojatnostnyj mashinnyj jeksperiment v priborostroenii [Probabilistic computer experiment in instrumentation] / V. K. Kruglikov. – L.: Mashinostroenie, Leningr. otd-nie, 1985. – 247 p. (in Russian).
4. Podlesnyj N. I. Special'nye metody identifikacii, proektirovanija i zhivuchest' sistem upravlenija [Special methods for the identification, design and management systems survivability] / N. I. Podlesnyj, A. A. Rassoha, S. P. Levkov. – K.: Vysshaja shkola, 1990. – 446 p. (in Russian).
5. Gregori J. Osnovy mnogopotochnogo, parallel'nogo i raspredelennogo programmirovanija [Tekst] / Jendrjus Gregori. – Addison-Wesley, 2003.
6. Yang L. Homeostatic and Tendency-based CPU Load Predictions / L. Yang, I. Foster, J. M. Schopf // International Parallel and Distributed Processing Symposium, 2003.
7. Smith W. Using run-time predictions to estimate queue wait times and improve scheduler performance. In Job Scheduling Strategies for Parallel Processing / W. Smith, V. Taylor and I. Foster., (D. G. Feitelson and L. Rudolph eds.). – Springer Verlag, 1999.
8. Brodskaja E.N. Metod molekuljarnoj dinamiki v fizicheskoj kolloidnoj himii [The method of molecular dynamics in physical colloidal chemistry]: Metodicheskoe posobie / E. N. Brodskaja, E. M. Piotrovskaja – NII himii SPbGU, 1999 (in Russian).
9. Allen M. P. Computer Simulation of Liquids / M. P. Allen, D. J. Tildesley. – Oxford University Press, 1987. – 406 p.
10. Vostokin S. V. Primenenie interpretatora scenarija GraphPlus dlja upravlenija raspredelennymi vychislenijami [Application shell script Graph Plus for distributed computing] / S. V. Vostokin // Izvestija SNC RAN. – M.: SNC RAN, 2005. – T. 7, № 1(13), pp. 138-142 (in Russian).

Соля В.П.
Навчально-науковий комплекс «Інститут прикладного системного аналізу»
Національного технічного університету України «Київський політехнічний інститут»

ПРИНЦИПИ ОРГАНІЗАЦІЇ РОЗПОДІЛЕНОГО ОБЧИСЛЕННЯ В КОМП'ЮТОРНИХ МЕРЕЖАХ

Анотація
В цій статті досліджуються принципи організації розподілених комп'ютерних обчислень в корпоративних мережах. Розв'язана задача розробки нового методу управління розподілених обчислювань в корпоративних мережах, який мінімізує час виконання обчислювань та покращує механізм розподілення задач всередині мережі.
Ключові слова: грід, координатор, диспетчер, агент, агресивність, надійність.

Соля В.П.
Учебно-научный комплекс «Институт прикладного системного анализа»
Национального технического университета Украины «Киевский политехнический институт»

ПРИНЦИПЫ ОРГАНИЗАЦИИ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ В КОМПЬЮТЕРНЫХ СЕТЯХ

Аннотация
В этой статье исследуются принципы организации распределённых компьютерных вычислений в корпоративных сетях. Решена задача разработки нового метода управления распределенных вычислений в корпоративных сетях, который минимизирует время выполнения вычислений и улучшает механизм распределение задач внутри сети.
Ключевые слова: грид, координатор, диспетчер, агрессивность, надежность.

УДК 338.487/.488

ОЦІНКА КОНКУРЕНТОПРИДАТНОСТІ РОЗРОБЛЕНИХ НОВИХ ПОСЛУГ БІЗНЕС-ГОТЕЛЮ «PRIME HOUSE» В МІСТІ КИЄВІ

Чуйко А.М., Чуйко М.М., Кобиф Н.Г.
Харківський торговельно-економічний інститут
Київського національного торговельно-економічного університету

Розглянуто доцільність і перспективність використання у готельно-ресторанному комплексі додаткових інноваційних послуг, зокрема: можливість відкривати двері номера за допомогою мобільного телефону (від компанії-розробника Open Ways), реалізація проекту «Віртуальний консьєрж-сервіс», інноваційне оформлення та обладнання конференц-залів (відокремлення у зоні для кава-брейків трьох функціональних ділянок), послуги приватного секретаря, перекладача тощо. Комплексний показник конкурентопридатності готельного продукту проектуємого ділового готелю «Prime House» категорії 4 зірки перевищує значення конкурентопридатності головного готелю-конкурента «Баккара» на 15,9%, що вказує на перспективність готельного продукту і його достатню конкурентопридатність. На основі проведених розрахунків побудовано модель конкурентопридатності готельного продукту.
Ключові слова: готель, готельно-ресторанний комплекс, комплексний показник, готельна послуга, конкурентні переваги, якість послуги.

Постановка проблеми. У наданні готельних послуг найважливішу роль відіграє питання їх якості та асортименту. Без якісного обслуговування готель не здатний досягти своїх основних цілей. Світова практика розвитку різних готельних корпорацій і ланцюгів, як правило, свідчить, що отримання прибутку є результатом високої якості обслуговування [1]. Отже, якість обслуговування в готельному господарстві – поняття комплексне, тісно пов'язане зі споживанням двох видів благ (товарів і послуг) і з двома видами відносин (матеріальних і нематеріальних). Нематеріальний елемент готельних послуг – це атмосфера, привабливість оточення, естетика, комфорт, відчуття, теплота обслуговування, доброзичливість, спокій і висока культура міжособистісного спілкування. До матеріальних належать номерний фонд, товарно-матеріальні ресурси і технологія надання послуг. Тому створення і впровадження нових інноваційних послуг,