ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2024, Vol. 63, No. 1, pp. 19–30. © Pleiades Publishing, Ltd., 2024.

INFORMATION PROCESSING AND IDENTIFICATION

Stochastic Models for the Time Complexity of Computing Tasks: I. Development Principles, Statistical Data Mining, and Identification Problems

A. V. Borisov[a],*, and A. V. Ivanov[a],**

[a] *Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, 119333 Russia*

**e-mail: ABorisov@frccsc.ru*

***e-mail: AIvanov@frccsc.ru*

Received June 7, 2023; revised August 16, 2023; accepted October 2, 2023

Abstract—This paper contains the first part of a study on the design of mathematical models for the execution time of user tasks on virtual calculating nodes. It is assumed that the execution time is a random value with the mean and variance depending on the node resources, task parameters, and the current characteristics of the node state. We discover the key features of the mean and variance functions and specify some of their particular cases. Both the mean and variance functions depend on the unknown parameters, and the design of the stochastic model for time complexity leads to parameter identification in the form of the generalized maximum likelihood estimates under heterogeneous statistical information. This paper also contains recommendations concerning the gathering and subsequent use of this information: the node testbed preparation, stress test planning, and processing of the data obtained. Specific illustrating examples of the proposed mathematical model will be presented in the subsequent parts of the study.

Keywords: stochastic model, parameter estimation, M-estimate, nonconvex and nonsmooth optimization problems, stress testing

DOI: 10.1134/S1064230724700035

INTRODUCTION

The rapid development of computer technology of varying performance, coupled with the widespread penetration of high-speed data networks, has led to the emergence of new service industries such as augmented and virtual reality [1], Industrial Internet [2], Internet of Things [3], Internet of Military Things [4], Internet of Vehicles [5], etc. The widespread distribution of 5G/6G mobile communication standards in the near future will provide users with the opportunity to access computing power of any level, regardless of their geographical location and without the need to own it, on the one hand, and will give computer network owners the ability to quickly distribute the load across geographically dispersed computing nodes, on the other hand. Such computer networks operate under the following conditions:

—high cost of creation and operation;

—strict demands of network users on the quality of services provided;

—strong competition between service provider networks.

These stringent requirements lead to the need to optimize the use of computer network resources.

The object of the study concerns the problems of distributing the computing load across nodes of different performance levels and allocating computing resources to different users. The ideal result from the provider's point of view is to provide each user with the minimal computing resources sufficient to solve his tasks, taking into account the mandatory compliance with the terms of the service level agreement (SLA). However, a number of fundamental problems prevent the achievement of this goal.

First, there is uncertainty regarding user requests to perform calculations. Despite the fact that the SLA concluded between the user and the provider determines the list of tasks whose solution is provided by the computer network, the intensity of the receipt of user tasks is unknown in advance. This means that the amount of dedicated resources required to meet the SLA may vary upward. In addition, the complexity of individual tasks also contains uncertainty. The reasons for this are not only differences in task volumes. Sometimes the execution time of the same task on the same hardware and software platform can differ sig-

nificantly for different input data. An example of this is the numerical solution of optimization problems [6, 7] under different initial conditions, using machine learning algorithms [8], compiling video landscapes of computer games [9], etc.

Second, most of the software used is proprietary, which excludes the possibility of its detailed algorithmic analysis in order to determine the complexity of performing a particular task. Regarding the procedure for completing tasks, it is possible to draw only the most general conclusions, based on knowledge of the functioning of similar open source programs, as well as algorithms for solving a particular class of problems. In some cases, there is direct and indirect statistical information about the execution time of tasks in the form of stress testing results.

Third, there is a known gap between user requirements for service quality as recorded in the SLAs and the control objects available to providers. The fact is that users determine the quality of their service through the maximum time it takes to complete tasks of a particular class. At the same time, to ensure the execution of user tasks, providers allocate a certain amount of computing resources: RAM and disk memory, processor cores, etc. The number of allocated resources affects the speed of task execution but does not determine the time of their completion due to the unknown time complexity.

The aim of this study is to propose stochastic models of the execution time of user tasks on computing nodes and methods for identifying the parameters of these models.

The article is organized as follows. Section 1 contains an informal description of the problems of estimating the probability of an SLA violation on a particular virtual computing node, which consists of exceeding the permissible execution time of a job. In this case, the scope of the user task is either known or the user profile is known, i.e., the probabilistic distribution of tasks of various sizes, characteristic for the given user.

Section 2 presents the basic principles for selecting task execution time models. An argument is given in favor of its probabilistic description, and useful properties that the selected models should have are proposed. In Section 3, the task completion time is proposed to be described by a family of random variables whose mean values $\mathcal{M}$ and variance $\mathcal{D}$ are functions of node resources, characteristics of the task being performed, and hardware and software environment variables. Similar to production functions [10], this section discusses the properties of functions $\mathcal{M}$ and $\mathcal{D}$ for different variables, and also some of their special cases are proposed. Here we can also find mathematically correct formulations and solutions to the problems of estimating the probability of an SLA violation.

In Section 4, it is proposed to use the weighted M-estimation methodology [11] to solve the problem of identifying the parameters of the proposed model of the complexity of user tasks. In this case, it is proposed to use one of the following convex functions as a loss function: quadratic, modulus, or convex piecewise linear.

Solving the problem of identifying parameters presupposes the availability of the necessary statistical information. Section 5 contains recommendations for its collection and processing. This information is accumulated in the process of stress testing, carried out on the hardware and software platform (hardware plus general system software (GSS)) on which the virtual computing node will subsequently operate. At the same time, real special software (SS) can be replaced by test software, and its brief classification is given in the section. In addition to the recorded execution time of tasks with various variable parameters, a list of recorded data is proposed, which can be useful for determining the variables of the hardware and software environment. The Conclusion contains brief conclusions on the model and prospects for further research.

1. DESCRIPTION OF THE PROBLEM

An informal description of the problem of characterizing the execution time of user tasks on computing nodes is presented. It is assumed that the node is virtual and is physically located on some computer hardware (CH): workstation, server, mainframe, etc. To ensure the functioning of the node, a certain number of processor cores, RAM, disk space, and other resources are allocated to the CH for their exclusive use. GSS and SS are deployed on the node. It is the SS that ensures the execution of user tasks arriving at the node.

All tasks that can be executed on the studied node are divided into types. Tasks are of the same type if they are performed by the same software using the same algorithms. For example, one type includes tasks for compressing video data, carried out by the same archiver program. If the same software is used to archive the text data, then this task belongs to a different type, since video and text information are com-

pressed using different algorithms. At the same time, tasks for compressing video information using different software tools also belong to different types, since they use different software.

Another example of tasks of various types are the tasks of loading information into a certain database (DB), on the one hand, and unloading the data obtained as a result of a certain request, on the other hand. In the following presentation it is assumed that problems of one fixed type are being studied.

Tasks belonging to the same type are different from each other. These differences are described by some vector of parameters. The simplest characteristic of a task is the amount of data processed during its execution; for example, the volume of video files to be compressed, the number of pictures to be recognized, the amount of information to be loaded into the database, etc. However, the job characteristics are not limited to the amount of data. For example, in a neural network training task, the following parameters can be used:

—number of layers in the network;

—maximum number of nodes in a layer;

—volume of the training bank;

—a parameter that determines the condition for stopping the learning process.

As noted earlier, the time required to complete individual tasks of one type or another contains uncertainty due to various reasons. The first one is the "closed" nature of the software's functioning. This type of uncertainty manifests itself in time fluctuations even when performing tasks with the same parameters.

The second reason lies in the uncertainty of the user's choice of one or another vector of task parameters and input data. This uncertainty can be partially reduced by forming a so-called *user profile* based on historical information about the requests of the given user to complete tasks of the type being studied. For example, a task to download films from a certain storage in 80% of cases is related to full-length films with a duration of 90 to 240 minutes; in 15% of cases, to short films with a duration of 10 to 30 minutes; and in 5% of cases, to documentaries lasting from 40 to 90 minutes. In this way, the user profile defines the conditional shares of the task with various parameters.

To ensure optimization of the use of CH resources in the presence of user SLAs, the following practical problems need to be solved.

Problem 1. For the fixed hardware resources of a node, the state of the GSS and SS, and a user task of a known type and volume, estimate the probability of violating the SLA—exceeding the maximum execution time of a task of this type.

Problem 2. For a known user profile with fixed hardware resources of the node, the state of the GSS and SS, estimate the probability of violating the SLA—exceeding the maximum execution time for a task of this type.

A formal description of the proposed models, the formulation of the problem of identifying their parameters, a mathematically correct formulation of the problems, and solution of the problems presented above are given in the following sections.

2. BASIC PRINCIPLES FOR CHOOSING MODELS

The choice of mathematical models for the execution time of tasks on computing nodes is fundamentally influenced by the presence of uncertainty in this time. This means that repeating the same job on the same computer node will require different amounts of time. As a mathematical apparatus that describes this phenomenon and allows solving problems of system analysis for it, we can use the theory of fuzzy sets [12], game/guaranteed approach [13, 14], etc. However, the most promising approach for this seems to be the apparatus of probability theory. Thus, the task completion time will be considered a random value $\tau(\omega)$, which means that the model that describes it belongs to the stochastic class.

A stochastic task execution time model should be selected based on the following principles.

1. *Adequacy*: the model specifies all those features of the actual execution time that are necessary to solve the full range of subsequent analysis problems, identify model parameters, and optimize them.

2. *Versatility*: The model provides the ability to adequately describe execution times for a wide range of computing nodes and user jobs.

3. *Simplicity*: the number of model parameters should be as minimal as possible while maintaining the required degree of adequacy and universality.

4. *Flexibility with respect to a priori information*: the model is adjusted in accordance with additional available information about the characteristics of CH, GSS, SS, and the presence of new or absence of any factors affecting the execution time of tasks of one type or another.

5. *Development*: for the selected models, there is an effective mathematical apparatus and algorithmic support for solving the entire range of applied problems related to the analysis of the studied time, identification of model parameters, and the possible subsequent optimization of the configuration of virtual computing nodes.

6. *Possibility of using simulators*: at the preliminary stage of identifying the model parameters when collecting statistical data on the task execution time, it is possible to use test software (TS) (*benchmarks*)—user load simulators and SS simulators.

3. STOCHASTIC MODELS FOR TIME COMPLEXITY: FORMAL DESCRIPTION, FORMULATION, AND SOLUTION OF THE ANALYSIS PROBLEM

According to the chosen stochastic approach, the execution time of a user task of some fixed type is proposed to be described by a random variable $\tau(\omega)$, having a finite mathematical expectation $\mathcal{M} \triangleq \mathrm{E}\{\tau\}$, and variance $\mathcal{D} \triangleq \mathrm{E}\{(\tau(\omega) - \mathcal{M})^2\}$. Both of these characteristics are unknown functions

$$\mathcal{M} = \mathcal{M}_z(x, y), \quad \mathcal{D} = \mathcal{D}_z(x, y),$$

where x is the vector of node resources, y is the vector of task parameters, and z is the vector of parameters of the hardware and software environment of the node. A detailed description of these vectors is given below.

The vector of variables $x = (x_1, \ldots, x_N)$ specifies the hardware and software composition of the virtual computing node, for example, x_1 is the number of processor cores, x_2 is the amount of available RAM, and x_3 is the amount of available disk space.

Note 1. The proposed model allows us to vary the set x depending on the composition and ability to configure the virtual node, adding some components to it, or excluding irrelevant ones. For example, the following ones can be added: x_4, the amount of disk space reserved for paging; x_5, the cache size; etc.

The vector $y = (y_1, \ldots, y_M)$ characterizes user jobs of a fixed type arriving at the given computing node. Components y_m for different types of tasks can have completely different meanings, but they all affect the execution time. For example, if the node is configured to perform scientific calculations, e.g., to numerically solve mathematical physics equations, then the following variant is possible: $M = 2$, y_1 is the number of time layers in the numerical solution, and y_2 is the number of spatial grid nodes in the numerical solution.

Another example: a node hosts a database, and the user task running on it involves the input/output of data and execution of queries to solve some information problem. In this case $M = 3$, y_1 is the volume of the input data in the task, y_2 is the volume of the output data in the task, and y_3 is the number of requests made to the database to perform informational tasks in a task.

Vector $z = (z_1, \ldots, z_K)$ contains the current parameters of the hardware and software environment of the computing node. For example, if a database is hosted on the given computing node, then the following characteristics can be the environment's parameters: z_1 is the maximum amount of RAM available on the node, z_2 is the current amount of useful data stored in the database, z_3 is the total current volume of service information stored in the database (transaction log, etc.), z_4 is the average amount of RAM of a computing node occupied by the GSS, etc.

Note 2. The following semantic differences exist between vectors x, y, and z. Vector x represents the control available to the provider. Vector y represents the control available to the user. Both x and y can vary independently of each other during stress testing and collection of statistical information for the subsequent identification of the model parameters. It should also be noted that due to the fact that different types of jobs can be executed on the same node, the dimension of vector x and the set of its possible values for the given node remains unchanged, in contrast to vector y, whose characteristics are different for tasks of different types.

Unlike x and y, parameters z of the hardware and software environment are not directly controlled by either the provider or the user. However, its components are available for direct or indirect observation during stress testing: information about it can be obtained either by executing service commands/queries or by analyzing system logs.

The set $\mathfrak{A}$ of valid values of arguments (x, y) is limited and, moreover, finite: components x_n, $n = \overline{1, N}$, and y_m, $m = \overline{1, M}$, can take values from some finite sets. Without loss of generality, it is assumed that the set of admissible vector values (x, y) is contained in the parallelepiped $\mathcal{U} = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, on which the functions $\mathcal{M}_z(x, y)$ and $\mathcal{D}_z(x, y)$ have the following properties.

1. *Nonnegativity:* for any (x, y, z), the following inequalities are valid:

$$\mathcal{M}_z(x, y) > 0, \quad \mathcal{D}_z(x, y) > 0.$$

These inequalities are obvious: the mathematical expectation and variance of any positive random variable with the first two moments finite are always positive.

2. *Local nonincreasing $\mathcal{M}_z(\cdot)$ on x:* for any fixed z, there is a subset $\mathcal{U}' \subseteq \mathcal{U}$ such that for any $(x', y^*), (x'', y^*) \in \mathcal{U}'$ such that $x' \le x''$, the following inequality is fulfilled component-by-component:

$$\mathcal{M}_z(x', y^*) \ge \mathcal{M}_z(x'', y^*).$$

This inequality means that in some subdomain $\mathcal{U}'$ an increase in the resources used leads to a decrease in the mean time for completing tasks; i.e., increasing the resources of a computing node makes sense.

3. *Nondecreasing $\mathcal{M}_z(\cdot)$ on y:* for any fixed z, the components of vector y can be defined so that for any $(x^*, y'), (x^*, y'') \in \mathfrak{A}$ such that $y' \le y''$, the following inequality is valid component-wise:

$$\mathcal{M}_z(x^*, y') \le \mathcal{M}_z(x^*, y'').$$

According to this property, vector y can be defined in such a way that all its components will have the meaning of the task volume, and with an increase in their values, the task completion time will increase on average.

4. *Continuity by y:* for any fixed x and z, functions $\mathcal{M}_z(x, y)$ and $\mathcal{D}_z(x, y)$ are continuous with respect to variable y. This property means that, with fixed resources, small variations in the size of a user task lead to small variations in the characteristics of its execution time.

5. *Convex $\mathcal{M}_z(\cdot)$ on y:* for any fixed z and for any $(x^*, y'), (x^*, y'') \in \mathcal{U}$ and $\lambda \in [0,1]$, the following inequality is valid:

$$\mathcal{M}_z(x^*, \lambda y' + (1-\lambda) y'') \le \lambda \mathcal{M}_z(x^*, y') + (1-\lambda) \mathcal{M}_z(x^*, y'').$$

According to the inequality, it means that with fixed resources, the mean job completion time as a function of the size of the user job grows linearly or faster.

Note 3. The locality of the nonincrease in the mean task execution time for resource variables (Property 2) appears exotic only at first glance. First, the analogous behavior of the mean execution time is observed when analyzing the functioning of real applications, such as databases. A specific example of this phenomenon will be given in the second part of this study. Second, the fact that the task completion time does not decrease with an increase in the amount of resources provided can be partly explained by the indirect restrictions and differences in the intensity of use of various types of resources. If we imagine the operation of the entire application as a collection of some data processing pipelines of varying intensity exploiting common node resources, then the following situation is quite possible. Among the mentioned conveyors, the least productive conveyor, the bottleneck, ultimately determines the intensity of the work of the entire node. Increasing some resources above a certain threshold can lead to a situation where the bottleneck will no longer cope with processing the increased data flow, and this data will be placed in a queue or simply uselessly lost, requiring its recreation. Moreover, due to the sharing of resources between pipelines, some of them will be taken away from the bottleneck by faster pipelines, further reducing its productivity.

Note 4. To illustrate Property 3, consider again the user task, which consists of numerically solving a certain equation of mathematical physics. Simply put, the task execution time is an increasing function of the number of grid nodes at which the solution to the equation needs to be calculated. The task parameters can be described as follows: y_1 is the grid step in spatial variable, y_2 is the grid time step, y_3 is the area of integration of the spatial variable, and y_4 is the area of integration over time.

However, such a parameterization will not satisfy property 3: the mean task execution time will not increase with increasing grid steps (parameters y_1 and y_2). The parameterization can be suitably modified so that property 3 is satisfied: the correct version is given in this section above, immediately after Note 1.

The listed properties of functions $\mathcal{M}_z(x,y)$ and $\mathcal{D}_z(x,y)$ impose certain restrictions on them. It is proposed to use the following dependencies in this study.

The first model assumes an exponential dependence of $\mathcal{M}_z$ and $\mathcal{D}_z$ on the resources–task volume pair (x, y) and independence of parameters z of the hardware and software environment

$$f_\varrho^1(x,y) = \alpha + \exp\left[\beta + \sum_{n=1}^{N}\gamma_n x_n + \sum_{m=1}^{M}\varepsilon_m y_m\right], \tag{3.1}$$

where $\varrho \triangleq \mathrm{vec}\left(\alpha,\beta,\{\gamma_n\}_1^N,\{\varepsilon_m\}_1^M\right)$ is a vector of unknown parameters to be subsequently identified separately for functions $\mathcal{M}_z$ and $\mathcal{D}_z$. Properties 1–5 of functions $\mathcal{M}_z$ and $\mathcal{D}_z$ dictate the presence of the following restrictions on the parameters' vectors:

(1) $\min\limits_{\{(x_n,y_m)\}}\left(\sum\limits_{n=1}^{N}\gamma_n x_n + \sum\limits_{m=1}^{M}\varepsilon_m y_m\right) + \beta - \ln\alpha \geq 0$ for functions $\mathcal{M}_z$ and $\mathcal{D}_z$ is the nonnegativity condition;

(2) $\gamma_n \leq 0$, $\overline{1,N}$ for function $\mathcal{M}_z$ is the condition of the local nonincrease in variables x_n, $\overline{1,N}$;

(3) $\varepsilon_m \geq 0$, $\overline{1,M}$ for function $\mathcal{M}_z$ is the condition of the local nondecrease in variables y_m, $\overline{1,M}$.

The exponential model can be used for the initial, so-called exploratory statistical analysis [15], necessary, for example, to determine the effective operation zone of a virtual node.

The second model assumes a power-law dependence on components (x, y) and independence of parameters z of the hardware and software environment. Thus, functions $\mathcal{M}_z$ and $\mathcal{D}_z$ of the second type have the form

$$f_\varrho^2(x,y) = \alpha + \beta\prod_{n=1}^{N}x_n^{-\gamma_n}\prod_{m=1}^{M}y_m^{\varepsilon_m}, \tag{3.2}$$

where $\varrho \triangleq \mathrm{vec}(\alpha,\beta,\{\gamma_n\}_1^N,\{\varepsilon_m\}_1^M)$ is the vector of unknown parameters to be subsequently identified separately for functions $\mathcal{M}_z$ and $\mathcal{D}_z$. Properties 1–5 of functions $\mathcal{M}_z$ and $\mathcal{D}_z$ dictate the presence of the following restrictions on the parameters' vectors, some of which can be set explicitly, and some of which depend on each specific identification task:

(4) $\alpha \geq 0$ and $\beta \geq 0$ for functions $\mathcal{M}_z$ and $\mathcal{D}_z$ is the nonnegativity condition;

(5) $\gamma_n \geq 0$, $n = \overline{1,N}$ for function $\mathcal{M}_z$ is the condition of the nonincrease in the variable x, $n = \overline{1,N}$;

(6) $\varepsilon_m \geq 0$, $m = \overline{1,M}$, for function $\mathcal{M}_z$ is the condition of the nondecrease in the variable y, $m = \overline{1,M}$.

The third model assumes a piecewise linear dependence on variable y. At the same time, subsets $\{U_j\}_{j=\overline{1,J}}$ of set $\mathcal{U}$, on which functions $\mathcal{M}_z(x,y)$ and $\mathcal{D}_z(x,y)$ are linear in variables y, form a partition $\mathcal{U}$: $U_i \cap U_j = \varnothing$ for $\forall\, i \neq j$ and $U_{j=1}^{J}U_j = \mathcal{U}$. Let us introduce indicator functions $\mathbf{I}_{U_j}(x,y)$ of sets $\{U_j\}_{j=\overline{1,J}}$:

$$\mathbf{I}_{U_j}(x,y) \triangleq \begin{cases} 1, & \text{if} \quad (x,y) \in U_j, \\ 0, & \text{if} \quad (x,y) \notin U_j. \end{cases}$$

To perform tasks of different sizes, a virtual computing node uses different amounts of resources. When tasks reach certain volumes, the node abruptly changes the procedure for processing them, while these changes are not directly observable. They can only be judged indirectly by analyzing the information stored in system logs. This jump is the reason for the change in the parameters of the linear dependence of the processing time on the job size. This phenomenon can be most clearly observed when varying the volume of the tasks and abrupt changes in the use of RAM. With small volumes, the entire task fits in the cache, the fastest-running area of RAM. Then, as the size increases, the task begins to fill the free part of RAM, and the processing speed decreases slightly. As soon as the amount of RAM is insufficient to fully accommodate the job, the paging mechanism is turned on, which dramatically increases the processing time.

Thus, functions $\mathcal{M}_z$ and $\mathcal{D}_z$ of the third type have the form

$$f_\varrho^3(x,y) = \alpha + \prod_{n=1}^{N}x_n^{-\beta_n}\sum_{j=1}^{J}\mathbf{I}_{U_j}(x,y)\left(\sum_{m=1}^{M}\gamma_{jm}y_m + \varepsilon_j\right), \tag{3.3}$$

where $\varrho \triangleq \mathrm{vec}\left(\alpha, \{\beta_n\}_1^N, \{\gamma_{jm}\}_{1,1}^{J,M}, \{\varepsilon_j\}_1^J\right)$ is the vector of unknown parameters to be subsequently identified separately for functions $\mathcal{M}_z$ and $\mathcal{D}_z$. It should be clarified that γ_{jm} are the coefficients of linear dependence on the variable y_m, $m = \overline{1, M}$, in area U_j, $j = \overline{1, J}$.

Properties 1–5 of functions $\mathcal{M}_z$ and $\mathcal{D}_z$ define the following restrictions on the estimated parameters:

(1) $\alpha \geq 0$ for functions $\mathcal{M}_z$ and $\mathcal{D}_z$ is the nonnegativity condition;

(2) $\beta_n \geq 0$, $n = \overline{1, N}$, for functions $\mathcal{M}_z$ is the local nonincreasing by variable x;

(3) $\gamma_{jm}^{\mathcal{M}} \geq 0$, $m = \overline{1, M}$, $j = \overline{1, J}$, is the nondecreasing $\mathcal{M}_z$ by variable y;

(4) joint restrictions on $\{\gamma_{jm}\}_{1,1}^{J,M}$ and $\{\varepsilon_j\}_1^J$, ensuring the continuity of functions $\mathcal{M}_z$ and $\mathcal{D}_z$ by variable y;

(5) joint restrictions on $\{\gamma_{jm}\}_{1,1}^{J,M}$ and $\{\varepsilon_j\}_1^J$, providing convexity of $\mathcal{M}_z$ by variable y.

Note 5. The models of the mean task completion time (3.2) and (3.3) allow the following economic interpretation. Let us assume that the algorithm that implements the solution of the user task belongs to the complexity class *P*. This means that the number of operations ensuring the execution of the task is limited from above by some power, in a particular case, by the linear function of the vector components y. Due to the use of the paging mechanism of pages, when there is insufficient RAM, the number of operations can increase, which is described using a power-law or convex piecewise linear function of the task's parameters. The task's execution intensity, which determines the number of algorithm operations performed on a computing node per unit of time, is close in meaning to the production function [10], in which the components of vector x act as resources. One of the popular functions, the Cobb-Douglas function, is the product of components x with some positive indicators. Knowing the total number of operations and the intensity of their execution, the task completion time is determined by their quotient. Models (3.2) and (3.3) represent the ratio of the upper estimates of the number of required operations to the intensity of their execution, described by the Cobb-Douglas function.

The proposed stochastic runtime models allow us to correctly formulate Problems 1 and 2 from the previous section as analysis problems and solve them.

Analysis task 1. On the virtual node, tasks of a fixed type are executed, and functions of the mean $\mathcal{M}_z(x, y)$ and variance $\mathcal{D}_z(x, y)$ time $\tau(\omega)$ of task completion are known. In addition, the triple node configuration–task parameters–hardware and software environment parameters (x, y, z) are fixed and known. Let $\overline{T}$ be the SLA parameter that determines the maximum allowable time for completing the user task y: estimate from above the probability of time $\tau(\omega)$ exceeding threshold $\overline{T}$.

The solution to the problem can be obtained using the Chebyshev inequality [16]:

$$\mathsf{P}\{\tau(\omega) > \overline{T}\} = \mathsf{P}\{\tau(\omega) - \mathcal{M}_z(x, y) > \overline{T} - \mathcal{M}_z(x, y)\}$$

$$\leq \frac{\mathcal{D}_z(x, y)}{\mathcal{D}_z(x, y) + (\overline{T} - \mathcal{M}_z(x, y))^2}. \tag{3.4}$$

Analysis task 2. The parameters of the configuration of node x are fixed and known and the parameters of the user task $Y(\omega)$ are random. However, the user task profile is known and specified in the form of a probability distribution $P(dy)$ of vector $Y(\omega)$. In addition, there is a known dependence connecting the value of parameter z on the pair (x, y): $z = z(x, y)$. Let $\overline{T}$ be the SLA parameter that determines the maximum execution time of a user task with parameters Y: estimate from above the probability of time $\tau(\omega)$ exceeding threshold $\overline{T}$.

The mean time to complete a user job $\overline{\mathcal{M}}$ is calculated using the total probability formula [14]:

$$\overline{\mathcal{M}} = \mathsf{E}\{\tau(\omega)\} = \mathsf{E}\{\mathsf{E}\{\tau(\omega) \mid \mathrm{Y}\}\} = \mathsf{E}\{\mathcal{M}_{z(x,Y)}(x, Y)\} = \int \mathcal{M}_{z(x,y)}(x, y)\, P(dy). \tag{3.5}$$

In a similar way, we can calculate the variance $\overline{\mathcal{D}}$:

$$\begin{aligned}\overline{\mathcal{D}} &= \mathsf{E}\left\{\left(\tau(\omega)-\overline{\mathcal{M}}\right)^2\right\}\\ &= \mathsf{E}\left\{\mathsf{E}\left\{\left(\left(\tau(\omega)-\mathcal{M}_z(x,Y)\right)+\left(\mathcal{M}_z(x,Y)-\overline{\mathcal{M}}\right)^2 \mid Y\right)\right\}\right\}\\ &= \mathsf{E}\left\{\mathsf{E}\left\{\left(\tau(\omega)-\mathcal{M}_z(x,Y)\right)^2 \mid Y\right\}\right\}+\mathsf{E}\left\{\mathsf{E}\left\{\left(\mathcal{M}_z(x,Y)-\overline{\mathcal{M}}\right)^2 \mid Y\right\}\right\}\\ &\quad + 2\mathsf{E}\left\{\mathsf{E}\left\{\left(\tau(\omega)-\mathcal{M}_z(x,Y)\right)\left(\mathcal{M}_z(x,Y)-\overline{\mathcal{M}}\right) \mid Y\right\}\right\}\\ &= \int \mathcal{D}_z(x,y)\,P(dy)+\int\left(\mathcal{M}_z(x,y)-\overline{\mathcal{M}}\right)^2 P(dy)\\ &= \int \mathcal{D}_z(x,y)\,P(dy)+\int \mathcal{M}_z^2(x,y)\,P(dy)-\left(\overline{\mathcal{M}}\right)^2.\end{aligned} \tag{3.6}$$

The solution to the problem can again be obtained using Chebyshev's inequality:

$$\mathsf{P}\{\tau(\omega)>\overline{T}\} \le \frac{\overline{\mathcal{D}}}{\overline{\mathcal{D}}+\left(\overline{T}-\overline{\mathcal{M}}\right)^2}. \tag{3.7}$$

Thus, the proposed stochastic model of the execution time of user tasks allows us to effectively solve an important analysis problem: estimating from above the probability of the SLA being violated.

Estimates (3.4) and (3.7), based on Chebyshev's inequalities, are very conservative. If there is additional information about the time distribution $\tau(\omega)$, then the proposed estimates can be significantly refined (see the works [17, 18] and links within them). Note that these assumptions are not overly onerous and are almost always met in practice.

The first assumption is that the distribution function $\tau(\omega)$ is concave on the positive semiaxis. In this case, the solution of the analysis task 1 can be obtained from Gauss's inequality:

$$\mathsf{P}\{\tau(\omega)>\overline{T}\} \le \begin{cases} 1-\dfrac{\overline{T}}{\sqrt{3(\mathcal{M}_z^2(x,y)+\mathcal{D}_z(x,y))}}, & \text{if} \quad 0\le\overline{T}\le\dfrac{2}{\sqrt{3}}\sqrt{\mathcal{M}_z^2(x,y)+\mathcal{D}_z(x,y)},\\ \dfrac{4(\mathcal{M}_z^2(x,y)+\mathcal{D}_z(x,y))}{9\overline{T}^2}, & \text{if} \quad \dfrac{2\sqrt{\mathcal{M}_z^2(x,y)+\mathcal{D}_z(x,y)}}{\sqrt{3}}<\overline{T}. \end{cases} \tag{3.8}$$

The second assumption is that $\tau(\omega)$ has a distribution density that is a unimodal function on the positive semiaxis. In this case, the solution of the analysis task 1 can be obtained from Cantelli's inequality:

$$\begin{aligned}&\mathsf{P}\{\tau(\omega)>\overline{T}\}\\ &\le \begin{cases} \dfrac{3\mathcal{D}_z(x,y)-\left(\overline{T}-\mathcal{M}_z(x,y)\right)^2}{3\mathcal{D}_z(x,y)+3\left(\overline{T}-\mathcal{M}_z(x,y)\right)^2}, & \text{if} \quad \mathcal{M}_z(x,y)\le\overline{T}\le\mathcal{M}_z(x,y)+\sqrt{\dfrac{5}{3}\mathcal{D}_z(x,y)},\\ \dfrac{4}{9}\dfrac{\mathcal{D}_z(x,y)}{\mathcal{D}_z(x,y)+\left(\overline{T}-\mathcal{M}_z(x,y)\right)^2}, & \text{if} \quad \mathcal{M}_z(x,y)+\sqrt{\dfrac{5}{3}\mathcal{D}_z(x,y)}<\overline{T}. \end{cases}\end{aligned} \tag{3.9}$$

Note 6. The type of functions $f_\varrho(\cdot)$ used to describe momentary characteristics of time $\tau(\omega)$ of the execution of a task is not limited to functions (3.1)–(3.3). The reasons for choosing these models in this study are as follows. The exponential dependence (3.1) can be used for the exploratory analysis of the behavior of the described moment characteristics and determination of the area (x, y) in which the node efficiently performs jobs of the specified type. The power model (3.2) has a successful economic interpretation. Model (3.3), which represents a power-law dependence on the resource parameters and a piecewise linear dependence on the volume of the task being performed, also has its strengths. First, it is simple and economical, as it includes a small number of parameters. Second, this model allows us to describe the abrupt change in the tasks' processing discipline depending on their volume.

4. PARAMETER IDENTIFICATION PROBLEMS

Let some fixed set $\{(x^r, y^r)\}_{r=\overline{1,R}}$ of variables (x, y), for which the set of stress testing of the studied computational node was carried out, be given. The result of testing is the set of vectors $\{\tilde{\mathcal{M}}^r, \tilde{\mathcal{D}}^r, \tilde{\mathcal{Z}}^r\}_{r=\overline{1,R}}$

obtained by processing the statistical information about the operation of the node for fixed values of the pairs (x^r, y^r). The first component $\tilde{\mathcal{M}}^r$ represents the sample mean time $\tau(\omega)$, the second is its variance, and the third component, possibly block component, consists of some known sample characteristics of the hardware-software environment corresponding to the pair (x^r, y^r). The set of positive weights $\{w_r\}_{r=\overline{1,R}}$, determining the individual significance of the results of each experiment, is also given.

The selection of models that describe the mathematical expectation and dispersion of the task completion time among possible functions (3.1)–(3.3), as well as the identification of their parameters, are carried out independently. Therefore, we will describe in detail the formulation of the identification problem only for the mathematical expectation function: the problem of identifying the parameters of the dispersion function will look absolutely similar. The value

$$\Delta_r^{\mathcal{M},i}(\varrho) \triangleq \tilde{\mathcal{M}}_r - f_\varrho^i(X_r, Y_r), \quad i = 1,2,3,$$

represents the error in estimating the mean job execution time when describing the configuration (x^r, y^r) of functions $f_\varrho^i(\cdot)$ calculated with the parameter values ϱ.

To compare the quality of models with different parameters, we will use the loss function $\pi(u): \mathbb{R} \to \mathbb{R}$, satisfying the following properties [11]: $\pi(0) = 0$, $\pi(u) \geq 0$ for all $u \in \mathbb{R}$, and function $\pi(u)$ is nonincreasing at $u < 0$ and nondecreasing at $u > 0$.

The problem of the optimal identification of a model for the mean task execution time lies in finding

$$(i^*, \varrho^*) \in \operatorname{Argmin}_{i,\varrho} \sum_{r=1}^{R} w_r \pi(\Delta_r^{\mathcal{M},i}(\varrho)). \tag{4.1}$$

The problem of the optimal identification of a model for the task execution time variance consists of defining

$$(i^*, \varrho^*) \in \operatorname{Argmin}_{i,\varrho} \sum_{r=1}^{R} w_r \pi(\Delta_r^{\mathcal{D},i}(\varrho)). \tag{4.2}$$

If the loss function is a quadratic function, i.e., $\pi(u) = u^2$, then (i^*, ϱ^*) is called the least squares estimate, if the absolute value is used, i.e., $\pi(u) = |u|$, then (i^*, ϱ^*) is an estimate of the smallest modules [19, 20]. In the more general case, when $\pi(u)$ is a piecewise linear function satisfying the properties presented above, (i^*, ϱ^*) is called a quantile estimate [21]. Obviously, (4.1) and (4.2) represent a weighted version of the problems of constructing M-estimates [19].

Note 7. Using different scales $\{w_r\}_{r=\overline{1,R}}$ makes practical sense. First, the accuracy of the individual values of the sample means and variances $(\tilde{\mathcal{M}}^r, \tilde{\mathcal{D}}^r)$ for different parameters (x^r, y^r) may be different, and this must be taken into account during the subsequent identification. This situation may arise in the case when, with a certain hardware and software composition of the node x^r, the task completion time y^r is quite large and there is no way to find a test sample of sufficient size for subsequent averaging and obtaining $(\tilde{\mathcal{M}}^r, \tilde{\mathcal{D}}^r)$ with acceptable accuracy. Second, in practice the following phenomenon is observed: in the case of scarce resources of x, there is a large variation in time $\tau(\omega)$, which leads to the low accuracy of sample moments $\tilde{\mathcal{M}}^r$ and $\tilde{\mathcal{D}}^r$. Since models $\mathcal{M}(\cdot)$ and $\mathcal{D}(\cdot)$ are constructed uniformly for all values of x, the inequalities of observations $(\tilde{\mathcal{M}}^r, \tilde{\mathcal{D}}^r)$ can be taken into account by choosing the appropriate weights w_r. Third, the owner of a computing node may have some additional a priori information about the user's preference to perform tasks with certain parameters more often than others. In relation to this, the task may arise to identify parameters in such a way that the model more accurately describes the execution time of the tasks preferred by the user, possibly to the detriment of the description of others.

5. STATISTICAL INFORMATION FOR IDENTIFYING MODELS: PRINCIPLES OF DATA MINING AND PROCESSING

The problem of identifying the parameters of tasks' time complexity model when they are executed on a certain virtual node is practical, and therefore the statistical data mining for its solution should be carried

out on the same hardware and software platform, which will then be used in reality. A difficulty can arise only when equipping a node with the planned SS: for various reasons it may not be available until the moment of actual operation. In this case, this SS should be replaced by a test software that is close in terms of purpose to the planned SS.

Currently, there is a fairly large selection of available TS, differing from each other in the following way:

—subject of testing (general computer performance, processor performance, graphics coprocessors, disk system, etc.),

—set of simulated applications (office programs, databases, scientific computing, etc.);

—level of cross-platform and readiness for stress testing (readiness out of the box/the need for preliminary compilation of code for the hardware and software platform being tested),

—level of service for collecting statistical information during stress testing (there is/is not a built-in ability to log the resources used),

—type of license (paid/free).

In the studied identification problem, the first two features are key when choosing a particular TS. The following types of TS allow us to simulate the load generated:

—office applications [22, 23],

—compilers, interpreters [24],

—data compression, including video [24],

—scientific computing (hydrodynamics, atmospheric modeling, molecular dynamics, etc.) [24],

—3D rendering [24],

—database transactions [25].

The use of TS instead of SS is acceptable at the initial stage of identification; however, final clarification of the model parameters is possible only based on the statistical information obtained during the operation of real SS. Further, regardless of the use of SS/TS, it should be taken into account that in order to solve the problems of the optimal identification of the mean time (4.1) and dispersion (4.2) of completing a task, it is necessary to repeat tests multiple times with the same values of the (x^r, y^r) pairs with the subsequent calculation of the corresponding sample moments $(\tilde{\mathcal{M}}^r, \tilde{\mathcal{D}}^r)$.

Beyond measuring time τ during stress testing, the data related to the state of the hardware and software environment should be recorded at some intervals, i.e., the information, whose processing will allow us to determine the components of vector z. Typically, the following information is recorded:

—percentage of CPU time used;

—amount of RAM;

—volume of recorded data;

—volume of data read;

—amount of page memory;

—amount of data written to the secondary page memory storage (swap);

—amount of data read from the swap, etc.

Typically, the frequency of the data recording is 1 s. For further processing of the obtained raw data, standard operations of averaging, finding the minimum/maximum/median values, etc., are applied to them. Additional information in the form of vector z allows us to build a partition $\mathcal{U}$ for subsequent application of the piecewise linear model (3.3). For example, using such data processing, we can determine the minimum size of a job from which the memory paging mechanism begins to be used, which radically slows down the execution of jobs.

In conclusion, it should be noted that the practical implementation of identifying the parameters of the described models is related to solving the following problems. The first one is more theoretical in nature and is related to the correct planning of a stress testing experiment [26, 27], ensuring the identifiability of the parameters [28], constructing consistent estimates for them, and characterization of the rate of convergence to the true values [29]. The second problem is largely algorithmic. As the identification problems (4.1) and (4.2) are optimization problems, in which the criteria can be not only nonconvex in terms of the variables being optimized but also nonsmooth. The set of strict statements and theoretically based algorithms for solving such optimization problems is quite scarce, which leads to the need to use heuristic and neural network algorithms. The problem lies in the rational choice of the optimization algorithm in accordance with the specifics of the criterion and type of constraints or the development of one's own version

of the algorithm. A fairly complete review and description of such optimization algorithms is given in the monograph [30].

CONCLUSIONS

The first part of this paper studies the theoretical aspect of constructing a mathematical model of the execution time of user tasks on a virtual computing node. An argument is given in favor of using a probabilistic apparatus to describe the fundamental property of the variability inherent in this time. Based on the need for subsequent identification of the parameters of the probabilistic time distribution model, it is proposed to limit ourselves to its first two moments. The arguments of the functions that describe the selected points include the characteristics of the resources, user tasks, and the hardware and software environment. This paper presents the properties of these functions, as well as some of their variants. Obviously, the functions contain unknown parameters that must be identified for each computing node and type of task based on the statistical data obtained as a result of stress tests. Among the various types of parameter estimates, it is proposed to use weighted M-estimates. The principles of collecting and processing statistical information based on the results of stress testing carried out to identify the parameters of the proposed model are presented.

Research in this area cannot be considered complete. The following directions are considered promising. First, parametric model identification is a fairly resource-intensive task that requires lengthy and varied stress testing. In relation to this, the issues of optimizing the test plan and adaptive refinement of the model parameters during the operation of the node seem to be very relevant. This area should also include the possible expansion and refinement of the class of functions that describe the momentary characteristics of the task execution time. Second, it is important to detail the model, allowing it to take into account some additional features of the functioning of virtual nodes: sharing of hardware resources by different virtual nodes, simultaneous access of different users to one virtual node, etc.

However, it is of paramount importance to check the possibility of using the proposed methodology for constructing models of the complexity of computational tasks of various types: implementation of information processing in databases, scientific computing, image processing, data compression, etc. This is the goal of the subsequent parts of this study.

FUNDING

This study was carried out using the infrastructure of the Center for Collective Use (CCU) “High-Performance Computing and Big Data” (CCU “Informatics”) of the Federal Research Center “Computer Science and Control,” Russian Academy of Sciences, Moscow.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. P. Sirohi, A. Agarwal, and P. Maheshwari, “A survey on augmented virtual reality: Applications and future directions,” in *Seventh International Conference on Information Technology Trends (ITT)* (Abu Dhabi, 2020), pp. 99–106.
2. P. Gupta, C. Krishna, R. Rajesh, et al., “Industrial Internet of Things in intelligent manufacturing: A review, approaches, opportunities, open challenges, and future directions,” Int. J. Interact. Des. Manuf. (2022). https://doi.org/10.1007/s12008-022-01075-w
3. I. Parveen, C.A., O. Anjali, and R. Sunder, “Internet of Things: A review on its applications,” in *Information and Communication Technology for Competitive Strategies (ICTCS, 2021), Lecture Notes in Networks and Systems* (2021), Vol. 400, pp. 123–134.
4. P. Fraga-Lamas, T. Fernández-Caramés, M. Suárez-Albela, L. Castedo, and M. González-López, “A review on Internet of Things for defense and public safety,” Sensors **16** (10), 1644 (2016).
5. M. Abu Talib, S. Abbas, Q. Nasir, and M. Mowakeh, “Systematic literature review on Internet-of-Vehicles communication security,” Int. J. Distrib. Sensor Networks **14**, 12 (2018).
6. J. Nocedal and S. Wright, *Numerical Optimization* (Springer, New York, 2006).
7. D. Bertsekas, *Convex Optimization Algorithms* (Athena Scientific, Belmont, 2015).
8. M. Kearns, *The Computational Complexity of Machine Learning* (MIT Press, Cambridge, 1990).

9. S. Teller and C. Séquin, "Visibility preprocessing for interactive walkthroughs," in *Proc. 18th Annual Conference on Computer Graphics and Interactive Techniques* (Las Vegas, 1991), pp. 61–70.
10. M. Intrilligator, *Mathematical Optimization and Economic Theory* (Prentice-Hall, Englewood Cliffs, N.J., 1971; Progress, Moscow, 1975).
11. P. Huber, *Robust Statistics* (Wiley, New York, 1981; Mir, Moscow, 1984).
12. A. Bronevich and A. Lepskii, *Fuzzy Models of Data Analysis and Decision Making* (VShE, Moscow, 2022) [in Russian].
13. A. Pankov and K. Siemenikhin, "Minimax estimation for singular linear multivariate models with mixed uncertainty," J. Multivar. Anal. **98** (1), 145–176 (2007).
14. A. Borisov, "Minimax estimation in regression under sample conformity constraints," Mathematics **9**, 1080 (2021).
15. J. Tukey, *Exploratory Data Analysis* (Addison-Wesley, Boston, 1977).
16. A. Shiryaev, *Probability* (Fizmatlit, Moscow, 1989) [in Russian].
17. K. Semenikhin, "Two-sided probability bound for a symmetric unimodal random variable," Autom. Remote Control **80** (3), 474–489 (2019).
18. R. Ion, C. Klaassen, and E. van der Heuvel, "Sharp inequalities of Bienaymé–Chebyshev and Gauß type for possibly asymmetric intervals around the mean," TEST **32**, 566–601 (2023). https://doi.org/10.1007/s11749-022-00844-9
19. E. Demidenko, *Linear and Nonlinear Regressions* (Finansy i statistika, Moscow, 1981) [in Russian].
20. V. Mudrov and V. Kushko, *The Method of Least Modules* (URSS, Moscow, 2022) [in Russian].
21. R. Koenker, *Quantile Regression* (Cambridge University Press, Cambridge, 2005).
22. https://bapco.com/products/sysmark-25/.
23. https://benchmarks.ul.com/pcmark10.
24. https://www.spec.org/cpu2017/.
25. https://www.tpc.org/tpce/
26. S. Ermakov, V. Kozlov, and A. Zhiglyavskii, *Mathematical Theory of Experimental Design* (Nauka, Moscow, 1983) [in Russian].
27. R. Black, *Managing the Testing Process*: *Practical Tools and Techniques for Managing Hardware and Software Testing* (Wiley, Indianapolis, 2009).
28. L. Ljung, *System Identification: Theory for the User* (Prentice Hall, Englewood Cliffs, N.J., 1987; Nauka, Moscow, 1991).
29. G. Ivchenko and Yu. Medvedev, *Mathematical Statistics* (URSS, Moscow, 2014) [in Russian].
30. A. Panteleev, D. Metlitskaya, and E. Aleshina, *Global Optimization Methods. Meta-Heuristic Strategies and Algorithms* (Vuzovskaya kniga, Moscow, 2013) [in Russian].