mathematics

Article

Stochastic Time Complexity Surfaces of Computing Node

Andrey Borisov * and Alexey Ivanov

Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44/2 Vavilova Str., 119333 Moscow, Russia; aivanov@frccsc.ru
* Correspondence: aborisov@frccsc.ru

Abstract: The paper is devoted to the formal description of the running time of the user task on some virtual nodes in the computing network. Based on the probability theory framework, this time represents a random value with a finite mean and variance. For any class of user task, these moments are the functions of the node resources, task numerical characteristics, and the parameters of the current node state. These functions of the vector arguments can be treated as some surfaces in the multidimensional Euclidean spaces, so the proposed models are called the stochastic time complexity surfaces. The paper also presents a class of functions suitable for the description of both the mean and variance. They contain unknown parameters which should be estimated. The article includes the statement of the parameter identification problem given the statistical results of the node stress testing, recommendations concerning the test planning, and preprocessing of the raw experiment data. To illustrate the performance of the proposed model, the authors design it for an actual database application—the prototype of the passengers' personal data anonymization system. Its application functions are classified into two user task classes: the data anonymization procedures and fulfillment of the statistical queries. The authors identify the stochastic time complexity surfaces for both task types. The additional testing experiments confirm the high performance of the suggested model and its applicability to the solution of the practical providers' problems.

Keywords: stochastic model; nonlinear regression; M-estimate; parameter identification; stress testing

MSC: 68Q87; 68M20

check for updates

Citation: Borisov, A.; Ivanov, A. Stochastic Time Complexity Surfaces of Computing Node. *Mathematics* **2023**, *11*, 4379. https://doi.org/10.3390/math11204379

Academic Editor: Davide Valenti

Received: 19 September 2023
Revised: 18 October 2023
Accepted: 19 October 2023
Published: 21 October 2023

1. Introduction

The forthcoming perspective of implementing the reliable high-speed 5G/6G cellular network standards provides permanent access to unbounded computational resources regardless of their possession and geographic location. This tendency appears in the areas of cloud [1,2], and edge [3,4] computing, computing power networks and synonims [5–9], and other systems of distributive computations. The providers have to develop the corresponding services under the conditions of

– the high creation, ownership, and operating costs,
– the high user requirements stated in the Service Level Agreements (SLA),
– the high requirements for the reaction time and efficiency of the computing resource management,
– the high competition between the rival providers.

All the conditions lead to the design of adequate and effective mathematical models of the computing networks, their components, and virtual computing nodes, in particular.

There is a fundamental gap between the user requirements fixed in the SLA and the instruments of their realization utilized by the service providers. Usually, the SLA determines the maximal admissible runtime for the user task. To meet the requirements, the provider has to create the virtual computing node [10–12] and allocate to it with a certain amount of hardware resources: processor cores, random access memory (RAM), and disk storage. The extensive enlargement of the hardware resources does not always lead to the required computing performance [13]. Hence, the adequate mathematical description of the relations between the runtime of the user task and the node resources represents one of the actual applied problems. Using this model, the provider would manage the resources efficiently, satisfying the Quality of Service (QoS) indicators and optimizing the overall resource utilization.

There is extensive literature on the general computational complexity theory [14–17] and the detailed complexity estimates for the most common computational algorithms. However, this information is insufficient to estimate the running time of each program realization of a known algorithm due to the specific organization of execution parallelism, usage of the cache, and RAM in general. The problem, which is the closest but not identical to the investigated one, is a characterization of the time complexity of the user computational tasks. According to the commonly accepted definition [18], the time complexity determines the algorithm running time as a function of the processed input length. The version of the problem statement is rather academic and does not consider the essential facts. The characteristic of the user task does not often represent a scalar but a vector. For example, the Hidden Markov Model (HMM) parameter identification problem [19] can be characterized by

– the dimensionality of the HMM,
– the dimensionality of the available observations,
– the length of the observations,
– the termination conditions of the iterative identification procedure.

In addition to the vector characterizing the user task, one should consider the number of hardware resources allocated for the virtual node. Therefore, in the presentation below, we refer to the functions of the vector argument, describing the dependency of the user task runtime and task parameters as the time complexity surfaces.

The next issue that complicates the design of a mathematical model is the heterogeneous uncertainty in the runtime.

1. The application software installed on the node to carry out the user task is usually proprietary. Hence, the information concerning the chosen algorithm and its program realization is unavailable to neither the user nor the service provider. Furthermore, almost any software uses an intricate combination of known and original algorithms, which complicates estimating the total runtime. Finally, the abstract estimates of the computational complexity of the algorithms do not consider specific conditions of the software functioning on the computational node equipped with some hardware resources. For example, the runtime increases significantly under a RAM shortage because of the intensive usage of the swapping mechanism. By contrast, the time may decrease when the node has multiple processor cores, and the program realization admits the effective algorithm parallelization. Finally, the user and service provider have only general information concerning the algorithms used in the node software.
2. Any special software that fulfills the user task can not work without a system platform, which includes: a host and guest operation systems (OS), hypervisors, various drivers, and firmware. This software is also generally proprietary, and the corresponding algorithms are unavailable for complexity analysis. The uncertainty caused by this type of software appears in the uncontrollable usage of the hardware and virtual resources generated by the periodic software update, backup operations, and optimization of the disc storage room. At the same time, the uncertainty can manifest positively via the effective processing parallelization in the databases [20,21] and acceleration of calculations by the speculative execution technique [22–24].
3. The whole variety of the hardware versions is an additional source of uncertainty in the time complexity of the user tasks. The variability of the execution time can be affected by all the various hardware models and chipsets.
4. The most investigated uncertainty type is caused by input data in the user tasks. First, it appears as a dispersion of the data volume under processing. Second, it would take

place in the output volume. Third, the execution time may vary due to the nature of the user task.

The main object of the recent investigations was regarding the uncertainty of the fourth type [25], although all these are hardly separable in practice. Uncertainty is one of the barriers to using the traditional mathematical frameworks of [26]. To take it into account, one can use a guaranteeing approach: for this type of algorithm, it is necessary to find the worst input data [27,28], leading to the maximum execution time. This way is practically ineffective because of its significant computational expenses and unnecessarily conservative results.

The probabilistic approach is more accepted: one can treat the task execution time as a random value, and its randomness is generated by the input data choice endowed by some probability distribution. Specifically, the average execution time [29] is characteristic of the time complexity. This paradigm has some weaknesses. First, the recent results do not consider the influence of the node hardware resources. Second, the usage of only averaged complexity does not provide the required precision of the solution for some practical problems. In fact, the knowledge of the averaged execution time and the Markov inequality [30] allows us to find the estimate from above for the probability of the SLA violation, i.e., the probability that random runtime exceeds some maximum allowable threshold. However, the estimate, based on the Markov inequality, is too conservative and might be useless in practice.

The aim of the paper is to present a new concept of the time complexity surface describing the running time of a user task at the virtual computing node, along with information technology of the parameter identification of this model.

The paper is organized as follows. Section 2 presents the actual problems arising before the computing network providers. The first problem consists of estimating the probability of the SLA requirement violation when the task runtime exceeds the maximal admissible threshold. The second (related) problem is to determine a conservative threshold value, such that the chance for the runtime to exceed it is not greater than some fixed confidence level. The problems illustrate the importance of efficient model choice. Section 3 is a key point of the paper and introduces a new stochastic model of the time complexity surfaces. We treat the running time of the user task as a random value with finite moments up to the second order. Both the average and the variance of the runtime are the functions of

– the resource characteristics of the computing node,
– the parameters of the user task,
– the vector of the current node state.

The section also presents the natural properties of the average and variance of the runtime and some variants to describe them. The proposed stochastic model permits the proper solution to the applied problems introduced in the previous section.

The average and variance surface functions contain unknown parameters. Section 4 presents the problem statements for the model parameter identification given the available statistical information. The section also includes recommendations concerning the organization of stress testing as an information base for parameter identification.

Section 5 is an illustrative one. It contains the extended numerical analysis of the time complexity surface model for the passengers' personal data anonymization system [31]. Section 5.1 describes the hardware and software platform of the virtual node. The most attention is paid to the description of the application software. Its functions are classified into two parts, which define the user tasks. Section 5.2 is devoted to the stochastic model of user task No. 1, realizing a complex procedure of data anonymization. The section describes this procedure, the obtained stochastic model, and an analysis of its performance. Section 5.3 contains the analogous information concerning user task No. 2, which represents the calculation of the sample statistical characteristics of the data. Section 5.4 presents the results of the final tests, which confirm the model applicability for the solution to the providers' problems, introduced in Section 2. It also includes remarks concerning the configuration of a virtual node for this tested software. Section 6 contains some concluding remarks and the prospects for further research.

2. An Applied Problem: Estimating the Chance of SLA Violation

To illustrate the importance of the mean and variance knowledge and to prepare the introduction of the stochastic time complexity surfaces, we investigate the following applied problems for the computing network providers.

Let us consider a computing node deployed on a fixed hardware, i.e., workstation, server, mainframe, etc. The node represents the virtual machine with the known hardware resources allocated to it in the monopolistic mode. The owner has installed both the system and application software to provide the execution of the user tasks on the node.

The user tasks processed on the node belong to a fixed class. Any task is characterized by some parameter vector y. Its components include a processed data volume, number of procedure iterations, quality characteristics of data processing, etc. We use the probabilistic paradigm and assume that any user task contains stochastic uncertainty. The running times $\tau(\omega|y)$ for all tasks are mutually independent and have partially known conditional distribution $P_\tau(d\omega|y)$ given the value y. The uncertainty generated by the variability of the task parameters has a probabilistic nature: each time, the user chooses the parameters $Y(\omega)$ independently according to the known distribution $P_Y(dy)$.

Usually, the SLA approves the maximal admissible execution time $\overline{T}$ for the task of the considered type. There are two applied problems connected with the violation of SLA.

Problem 1. *Given the threshold value $\overline{T}$ it is necessary to estimate from above the probability for the task runtime $\tau(\omega)$ to exceed $\overline{T}$, i.e., $\mathcal{P}\{\tau(\omega) > \overline{T}\}$.*

Problem 2. *Given the admissible SLA confidence level $\overline{P}$ it is necessary to find a threshold $\overline{T}$, which guarantees that $\mathcal{P}\{\tau(\omega) > \overline{T}\} \leqslant \overline{P}$.*

The solution to Problem 1 is necessary when the threshold $\overline{T}$ is already fixed. In this case, the provider has to estimate from above the probability of violating the SLA condition and the probable penalties.

The solution to Problem 2 is necessary when the provider fixes the allowable probability $\overline{P}$ of the SLA violation, so he/she needs to determine the corresponding threshold $\overline{T}$.

Below, we consider the following a priori information concerning the distribution $P_\tau(d\omega|y)$. All solutions, presented in this section, are based on the inequalities accumulated in [30].

(A) The available information represents the dependency $\mathcal{M}(y)$ of the average runtime on the task parameters, i.e., the conditional expectation

$$\mathcal{M}(y) = \mathsf{E}\{\tau(\omega)|Y(\omega) = y\}.$$

(B) The available information includes the dependency of the mean $\mathcal{M}(y)$ and variance $\mathcal{D}(y)$ of $\tau(\omega)$ given the task parameters:

$$\mathcal{M}(y) = \mathsf{E}\{\tau(\omega)|Y(\omega) = y\}, \qquad \mathcal{D}(y) = \mathsf{E}\Big\{(\tau(\omega) - \mathcal{M}(y))^2|Y(\omega) = y\Big\}.$$

(C) Both $\mathcal{M}(y)$ and $\mathcal{D}(y)$ are given; for all y the distribution of τ is unimodal on $[0, +\infty)$.

(D) Both $\mathcal{M}(y)$ and $\mathcal{D}(y)$ are given; for all y the distribution of τ is concave on $[0, +\infty)$.

We present the solution for Problems 1 and 2 under conditions A–D in a unified manner. First, we solve Problem 1 given the condition $Y(\omega) = y$ and apply the formula of the total probability. Second, we solve Problem 2 using the solution to Problem 1.

Condition A is typical for the traditional average-case complexity approach. Using the formula of the total probability, we calculate the mean runtime corresponding to the task parameter distribution $P_Y(dy)$

$$\mathbf{M} \triangleq \mathsf{E}\{\tau(\omega)\} = \int_{[0,+\infty)} \mathsf{E}\{\tau(\omega)|Y(\omega) = y\} P_Y(dy) = \int_{[0,+\infty)} \mathcal{M}(y) P_Y(dy). \quad (1)$$

The solution to Problem 1 represents the usage of the Markov inequality

$$\mathcal{P}\{\tau(\omega) > \overline{T}\} \leqslant \frac{\mathbf{M}}{\overline{T}}. \quad (2)$$

Note, that in the case $\overline{T} \leqslant \mathbf{M}$ inequality (2) becomes useless:

$$\mathcal{P}\{\tau(\omega) > \overline{T}\} \leqslant 1,$$

so further we consider only the case $\overline{T} \geqslant \mathbf{M}$.

From inequality (2) it follows, that

$$\overline{P} = \mathcal{P}\{\tau(\omega) > \overline{T}\} \leqslant \frac{\mathbf{M}}{\overline{T}},$$

and the solution to Problem 2 is

$$\overline{T} = \frac{\mathbf{M}}{\overline{P}}. \quad (3)$$

To solve the problems under conditions B–D, we have to calculate the unconditional variance of $\tau(\omega)$

$$\begin{aligned} \mathbf{D} = \mathsf{E}\Big\{(\tau(\omega) - \mathbf{M})^2\Big\} &= \mathsf{E}\Big\{\mathsf{E}\Big\{((\tau(\omega) - \mathcal{M}(Y)) + (\mathcal{M}(Y) - \mathbf{M}))^2 | Y\Big\}\Big\} \\ &= \mathsf{E}\Big\{\mathsf{E}\Big\{(\tau(\omega) - \mathcal{M}(Y))^2 | Y\Big\}\Big\} + \mathsf{E}\Big\{\mathsf{E}\Big\{(\mathcal{M}(Y) - \mathbf{M})^2 | Y\Big\}\Big\} \\ &\quad + 2\mathsf{E}\Big\{\mathsf{E}\Big\{\Big(\tau(\omega) - \mathcal{M}(Y)\Big)(\mathcal{M}(Y) - \mathbf{M}) | Y\Big\}\Big\} \\ &= \int \mathcal{D}(y) P_Y(dy) + \int (\mathcal{M}(y) - \mathbf{M})^2 P_Y(dy) \\ &= \int \mathcal{D}(y) P_Y(dy) + \int \mathcal{M}^2(y) P_Y(dy) - \mathbf{M}^2. \quad (4) \end{aligned}$$

If Condition B is valid, the solution to Problem 1 can be obtained from one-sided Cantelli's inequality

$$\mathcal{P}\{\tau(\omega) > \overline{T}\} \leqslant \frac{\mathbf{D}}{\mathbf{D} + (\overline{T} - \mathbf{M})^2}, \quad (5)$$

and the solution to Problem 2 follows from the inequality $\overline{P} \leqslant \frac{\mathbf{D}}{\mathbf{D} + (\overline{T} - \mathbf{M})^2}$:

$$\overline{T} = \mathbf{M} + \sqrt{\mathbf{D}\left(\frac{1}{\overline{P}} - 1\right)}. \quad (6)$$

In some cases, the additional information concerning the distribution of $\tau(\omega)$ can significantly improve the performance of the solutions to Problem 1 and Problem 2. If Condition C is valid, the solution to Problem 1 can be obtained from a corollary of Cantelli's inequality

$$\mathcal{P}\{\tau(\omega) > \overline{T}\} \leqslant \begin{cases} \frac{3\mathbf{D} - (\overline{T} - \mathbf{M})^2}{3\left(\mathbf{D} + (\overline{T} - \mathbf{M})^2\right)}, & \text{if } \mathbf{M} \leqslant \overline{T} \leqslant \mathbf{M} + \sqrt{\frac{5}{3}\mathbf{D}}, \\ \frac{4}{9} \frac{\mathbf{D}}{\mathbf{D} + (\overline{T} - \mathbf{M})^2}, & \text{if } \mathbf{M} + \sqrt{\frac{5}{3}\mathbf{D}} < \overline{T}, \end{cases} \quad (7)$$

and the solution to Problem 2 is expressed by the formula

$$\overline{T} = \begin{cases} \mathbf{M} + \sqrt{\frac{3\mathbf{D}(1-\overline{P})}{3\overline{P}+1}}, \text{ if } \mathbf{M} \leqslant \overline{T} \leqslant \mathbf{M} + \sqrt{\frac{5}{3}\mathbf{D}}, \\ \mathbf{M} + \sqrt{\mathbf{D}\left(\frac{4}{9\overline{P}} - 1\right)}, \text{ if } \mathbf{M} + \sqrt{\frac{5}{3}\mathbf{D}} < \overline{T}. \end{cases} \tag{8}$$

If, in turn, Condition D is valid, the solution to Problem 1 can be obtained from the one-sided Gauß inequality

$$\mathcal{P}\{\tau(\omega) > \overline{T}\} \leqslant \begin{cases} 1 - \frac{\overline{T}}{\sqrt{3(\mathbf{M}^2+\mathbf{D})}}, \text{ if } 0 \leqslant \overline{T} \leqslant \frac{2}{\sqrt{3}}\sqrt{\mathbf{M}^2 + \mathbf{D}}, \\ \frac{4(\mathbf{M}^2+\mathbf{D})}{9\overline{T}^2}, \text{ if } \frac{2}{\sqrt{3}}\sqrt{\mathbf{M}^2 + \mathbf{D}} < \overline{T}, \end{cases} \tag{9}$$

and the solution to Problem 2 is expressed by the formula

$$\overline{T} = \begin{cases} (1 - \overline{P})\sqrt{3(\mathbf{M}^2 + \mathbf{D})}, \text{ if } 0 \leqslant \overline{T} \leqslant \frac{2}{\sqrt{3}}\sqrt{\mathbf{M}^2 + \mathbf{D}}, \\ \frac{2}{3}\sqrt{\frac{\mathbf{M}^2+\mathbf{D}}{\overline{P}}}, \text{ if } \frac{2}{\sqrt{3}}\sqrt{\mathbf{M}^2 + \mathbf{D}} < \overline{T}. \end{cases} \tag{10}$$

To illustrate both the advantages and limitations of a priori information B–D in comparison with traditional average case A, we consider two numerical problems.

Problem 3. *Let the mean and variance of* $\tau(\omega)$ *be known:* $\mathbf{M} = 1$, $\mathbf{D} = 0.01$. *It is necessary to solve Problem 1 for the threshold value* $\overline{T} = 1.5$ *and Problem 2 for the confidence level* $\overline{P} = 0.01$. *One should compare the solutions calculated given a priori information A–D with the case of the test Erlang distribution, which has the parameters* $\lambda = 100$ *and* $n = 100$.

Problem 4. *Under the conditions of Problem 3, the variance value* $\mathbf{D} = 0.0625$. *The task again is to solve Problems 1 and 2 and compare the results obtained for a priori information A–D with the case of the test Erlang distribution, which has the parameters* $\lambda = 16$ *and* $n = 16$.

Note that the mathematical expectation equal to one is convenient to use as a unit for measurements.

Table 1 contains the numerical results of Problems 3 and 4.

Table 1. Solution data for Problems 3 and 4.

Problem 3: $\mathbf{D} = 0.01$	Cond. A	Cond. B	Cond. C	Cond. D	Erl (100,100)
$\overline{P}$	0.667	0.038	0.017	0.200	3×10^{-7}
$\overline{T}$	100	1.995	1.659	6.700	1.233
Problem 4: $\mathbf{D} = 0.0625$	Cond. A	Cond. B	Cond. C	Cond. D	Erl (16,16)
$\overline{P}$	0.667	0.200	0.089	0.210	0.023
$\overline{T}$	100	3.487	2.648	6.872	1.582

By analyzing them, we can make the following conclusions. First, the solutions to Problems 3 and 4 have inappropriate levels of precision in the case of a priori information A. The point is that Markov inequality, used in this case, involves the average value $\mathbf{M}$ of the running time $\tau(\omega)$ only and provides results which are too conservative. Actually, the estimate from the SLA violation probability $\mathcal{P}\{\tau(\omega) > \overline{T}\}$ above, calculated in case A, is equal to 0.667, though it can be reduced to 0.038 in case B, and even to 0.017 in case C. In turn, for the test Erlang distribution, this value vanishes. At the same time, the threshold $\overline{T}$ calculated via the Markov inequality is 100 times greater than the average time **M**. If the provider includes this oversized time-out in the SLA, this choice could repel potential clients.

Second, the precision of the solution to Problems 3 and 4 in case D occupies the intermediate position between the ones obtained in cases A and B. The most precise estimates can be obtained under a priori information C, and calculated by Cantelli's inequality. It is not surprising that the estimates use the most advanced information including mean and variance knowledge, and the fact of the distribution unimodality. In summary, cases B and C of a priori information are preferable for the solution to Problems 3 and 4.

Third, the difference between Problem 3 and 4 is the ratio $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}}$: $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}} = 0.1$ for Problem 3 and $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}}$: $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}} = 0.25$ for Problem 4. Comparing the results, one can conclude that the precision of the solution decreases when $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}}$ increases. If the value of the ratio is significant, the estimate $\overline{P}$ of the SLA violation could be too pessimistic, meanwhile the calculated value $\overline{T}$ of the SLA threshold could look unsavory for potential clients.

In spite of the apparent simplicity of Problems 3 and 4, their solution has significant practical value. They establish the formal theoretical basis for the efficient allocation of the hardware resources to the computing nodes to meet user requirements. At the same time, the solution is a hint for the correct choice of the maximally admissible task running time during the preparation of SLA materials: the promised maximal time should meet the real potential of the computing node.

Obviously, both the mean and variance of the task running time are affected by many factors, and the next section introduces a mathematical model to describe this dependency.

3. Proposed Model: Stochastic Time Complexity Surfaces

According to the chosen probabilistic paradigm, we treat the runtime of a fixed type user task as a random value $\tau(\omega)$ with a finite mathematical expectation $\mathcal{M} \triangleq \mathsf{E}\{\tau(\omega)\}$ and variance $\mathcal{D} \triangleq \mathsf{E}\{(\tau(\omega) - \mathcal{M})^2\}$. Both these moment characteristics are unknown functions of the arguments (X, Y) and parameters Z:

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

The vector $x = (x_1, \dots, x_N)'$ presents the available resources of the node, e.g.,

- x_1 is a number of processor cores,
- x_2 is an available RAM amount,
- x_3 is an available storage amount,
- x_4 is a storage amount reserved for swapping,
- x_5 is a cache volume.

The vector $y = (y_1, \dots, y_M)'$ characterizes individual user tasks, used for processing on the computing node. Both the dimensionality M and the meaning of each component y_m vary for different types of user tasks. For example, if the node is configured for scientific calculations, say, for the numerical solution to partial differential equations, the vector y could have the following structure:

- $M = 2$,
- y_1 is an amount of the time layers of the solution,
- y_2 is an amount of the mesh nodes in one time layer.

Another example: if the node is created as a database server, and the user task assumes the data input/output and execution of a query, then

- $M = 3$,
- y_1 is a volume of the input data,
- y_2 is a volume of the output data,
- y_3 is an amount of the queries in the task.

The vector $z = (z_1, \dots, z_K)'$ accumulates the parameters of the current node state. For example, for the node-database server, these parameters represent

- z_1 is a maximal available RAM volume,
- z_2 is a current volume of the database,
- z_3 is a volume of the service data currently stored in the database: transaction log, etc.,
- z_4 is an average RAM volume, assigned for the system software,
- z_5 is an average RAM volume, assigned for the task execution.

Usually, researchers consider the task running time as a function of the single argument, which is the length of the input under processing and does not consider the node hardware configuration. Choosing the probabilistic approach, we suggest two moments, the mean and the variance, to characterize the random nature of the runtime. Moreover, these moments are the functions of the *vector arguments* x and y, and this makes them similar to *some surfaces* in a multidimensional space. That is why below we refer to this model as *the stochastic time complexity surfaces*.

The sense of the vectors x, y, and z is quite different. The vector x represents "a control" available for the provider. The vector y is "a control" available for the user. Both of x and y vary independently during the stress testing or routine operation. By contrast, the components of z can be determined neither by the user nor the provider but can be observed directly or indirectly via the service queries or the log analysis.

The set $\mathfrak{A}$ of admissible values (x, y) is bounded and, moreover, finite: the components $x_n,\ n = \overline{1, N}$ and $y_m,\ m = \overline{1, M}$ can be chosen only from some finite sets. Without the loss of generality, we suppose that the admissible set lies in a parallelepiped $\mathcal{U} = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, and on $\mathcal{U}$ the functions $\mathcal{M}_z(x, y)$ and $\mathcal{D}_z(x, y)$ have the following properties.

(i) *Positivity:*

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

for any (x, y, z). These inequalities are obvious: both the mean and variance of an arbitrary nonnegative nonsingular random value are positive.

(ii) *Local non-increase of $\mathcal{M}_z(\cdot)$ in x:* for any fixed z there exists a subset $\mathcal{U}' \subseteq \mathcal{U}$, such that for any $(x', y^*),\ (x'', y^*) \in \mathcal{U}' : x' \leqslant x''$ component-wise, and the inequality $\mathcal{M}_z(x', y^*) \geqslant \mathcal{M}_z(x'', y^*)$ is true.
The inequality implies the existence of a subset $\mathcal{U}'$, where the increase in the node resources leads to the decrease in the mean task running time. It guarantees that a resource extension makes sense.

(iii) *Non-decrease of $\mathcal{M}_z(\cdot)$ in y:* fixing any z the components of y can be determined in such a way that for all $(x^*, y'),\ (x^*, y'') \in \mathcal{U}$: $y' \leqslant y''$ component-wise, and the inequality $\mathcal{M}_z(x^*, y') \leqslant \mathcal{M}_z(x^*, y'')$ is true.
The property implies that y can be organized, so that all its components play the role of the task volume, and its increase leads to the one of the mean task runtime.

(iv) *Continuity in y:* for any fixed x and z the functions $\mathcal{M}_z(x, y)$ and $\mathcal{D}_z(x, y)$ are continuous in y. The property means that small perturbations of the user task parameters generate small variations in the moment characteristics of the task runtime.

The local character of the $\mathcal{M}_z(\cdot)$ non-increase in x (Property ii) has a very transparent explanation. First, this feature is known for the queueing systems (see an example of a congested network [13]). Second, the synchronous growth of the node resources and non-decrease, even an increase in the running time, can be partially explained by the distinction of the usage intensity of the different resources. We consider the whole functioning of the application as a collection of conveyors, which have different rates and share the common node resources. Among the conveyors, there is one with the lowest rate, called a bottleneck. When some resource consumption exceeds a threshold, the bottleneck stops the regular service of the increased input data flow, which leads to their queuing or even losses with the necessity of reprocessing. Moreover, the part of resources can be taken by more productive conveyors from the bottleneck, which reduces the processing rate of the latter one even more.

To illustrate Property iii, we consider a user task representing the numerical solution to a partial differential equation. Roughly speaking, the mean of the task running time is an increasing function of the total amount of the mesh covering the solution area. We can form the set of the task parameters in the following way:

- y_1 is a step of the space variable,
- y_2 is a step of the time,
- y_3 is a lower boundary of the space area,
- y_4 is an upper boundary of the space area,
- y_5 is an upper limit of the time.

The proposed parameter collection, just y_1 and y_2, violates Property iii. However, previously mentioned in the paper, we present a correct set of the parameters in this task, which preserves Property iii.

To describe both the mean $\mathcal{M}_z(\cdot)$ and variance $\mathcal{D}_z(\cdot)$ we choose a unified function of the following form

$$g(x, y, \rho) = a + b\left\{\prod_{n=1}^{N} x_n^{-c_n}(1 + d_n x_n^{e_n})\right\}\left[\prod_{m=1}^{M} y_m^{f_m}\right], \tag{11}$$

where

$$\rho \triangleq \mathrm{vec}\Big(a,\ b,\ \{c_n\}_{n=\overline{1,N}},\ \{d_n\}_{n=\overline{1,N}},\ \{e_n\}_{n=\overline{1,N}},\ \{f_m\}_{m=\overline{1,M}}\Big)$$

is a $(3N + M + 2)$ -dimensional vector of nonnegative model parameters, which should be identified. There is an additional constraint, connecting the parameters $\{c_n\}_{n=\overline{1,N}}$ and $\{e_n\}_{n=\overline{1,N}}$:

$$c_n \leqslant e_n, \quad n = \overline{1, N}. \tag{12}$$

First, we argue the chosen class of functions (11) describes the mean task runtime. In general, the time could be determined as a ratio of the operation number V and the productivity rate R: $T = V/R$. In some cases (see, e.g., the neural network learning process [32]), the operation number is a simple product of the y components, i.e., $V = \prod_{m=1}^{m} y_m$. The square bracket $\prod_{m=1}^{M} y_m^{f_m}$ in (11) generalizes the coordinate-wise product and corresponds to the fact that most of the contemporary algorithms realized in computers have a power-like complexity [27].

Further, the product in the figure brackets of (11) can be rewritten in the form $\prod_{n=1}^{N}\Big(d_n x_n^{-(e_n - c_n)} + x_n^{-c_n}\Big)$, where $e_n - c_n \geqslant 0$. Note, that the product $\prod_{n=1}^{N} x_n^{c_n}$ is the well-known Kobb-Douglas productivity function used here as a productivity rate. Hence, the formula

$$T = \left\{\prod_{n=1}^{N} x_n^{-c_n}\right\}\left[\prod_{m=1}^{M} y_m^{f_m}\right] \tag{13}$$

corresponds to the ratio $T = V/R$ completely, and also represents a specific case of (11) with the following parameter values: $a = 0$, $b = 1$, $d_n \equiv 0$.

Each multiplier in the figure brackets of (11) can be rewritten in the form $(x_n^{-c_n} + d_n x_n^{e_n - c_n})$. Note that due to (12), the power of the second summand is nonnegative. This fact guarantees Property ii: each multiplier in the figure brackets decreases locally for a while and then starts to increase with asymptotic $d_n x_n^{e_n - c_n}$.

As for the choice of (11) to describe the variance, the proposed formula contradicts neither Property i nor iv. Moreover,

$$T^2 = \left\{\prod_{n=1}^{N} x_n^{-2c_n}\right\}\left[\prod_{m=1}^{M} y_m^{2f_m}\right]$$

and belongs to the class of (11) with different parameters.

Model (11) can be extended up to the piecewise case. To do this, we suppose the existence of the partition family $\{\mathcal{U}_j(z)\}_{j=\overline{1,J}}$ of the set $\mathcal{U}$, parametrized by the parameter vector z. The class of admissible functions describing both $\mathcal{M}$ and $\mathcal{D}$ takes the form

$$h_z(x,y,\rho) = \sum_{j=1}^{J} \mathbf{I}_{U_j(z)}(x,y)\left(a_j + b_j \prod_{n=1}^{N} x_n^{-c_{nj}}\left(1 + d_{nj}x_n^{e_{nj}}\right)\prod_{m=1}^{M} y_m^{f_{mj}}\right), \tag{14}$$

where $\mathbf{I}_{U_j(z)}(x,y)$ is the indicator function of the set $U_j(z)$. The $J(3N+M+2)$-dimensional vector of the nonnegative model parameters

$$\rho \triangleq \operatorname{vec}\left(\{a_j\}_{j=\overline{1,J}}, \{b_j\}_{j=\overline{1,J}}, \{c_{nj}\}_{\substack{n=\overline{1,N},\\ j=\overline{1,J}}}, \{d_{nj}\}_{\substack{n=\overline{1,N},\\ j=\overline{1,J}}}, \{e_{nj}\}_{\substack{n=\overline{1,N},\\ j=\overline{1,J}}}, \{f_{mj}\}_{\substack{m=\overline{1,M},\\ j=\overline{1,J}}}\right)$$

with additional constraints

$$c_{nj} \leqslant e_{nj}, \quad n = \overline{1,N}, \quad j = \overline{1,J} \tag{15}$$

should be identified. Note, that the partition $\{\mathcal{U}_j(z)\}_{j=\overline{1,J}}$ of the set $\mathcal{U}$ forms additional constraints of the vector ρ: it should satisfy Property iv.

We illustrate the sense of the parameters z and their role in the formation of the partition family $\{\mathcal{U}_j(z)\}_{j=\overline{1,J}}$ in the next section.

4. Parameter Identification of Stochastic Time Complexity Surfaces

In spite of the similarity of models (11) and (14), processes of their parameter identification have significant distinctions: the structure of the statistical data collected during the stress testing differs for these cases, and the parameter identification procedures themselves are different for the models.

First, we formulate the parameter identification problem for the model (11). Let $\{(x^r, y^r)\}_{r=\overline{1,R}}$ be a fixed collection of the variables (x,y), chosen for the stress testing. Its result is processed into the set $\{\widetilde{\mathcal{M}}^r, \widetilde{\mathcal{D}}^r\}_{r=\overline{1,R}}$ of the sample moments (i.e., means and variances) for the runtime corresponding to the task with the parameters y^r fulfilled on the node with the resources x^r. Let also the set $\{w_r\}_{r=\overline{1,R}}$ of positive observation weights, expressing the individual significance of the tuple $(x^r, y^r, \widetilde{\mathcal{M}}^r, \widetilde{\mathcal{D}}^r)$ be given. The value

$$\Delta_r^{\mathcal{M}}(\rho) \triangleq \widetilde{\mathcal{M}}^r - g(x^r, y^r, \rho)$$

is the residual of the observed mean $\widetilde{\mathcal{M}}^r$ corresponding to the pair "resource parameters–task parameters" (x^r, y^r), described by the model (11) with the parameters ρ. Similarly, the value

$$\Delta_r^{\mathcal{D}}(\rho) \triangleq \widetilde{\mathcal{D}}^r - g(x^r, y^r, \rho)$$

is the residual of the observed variance $\widetilde{\mathcal{D}}^r$ corresponding to the pair (x^r, y^r), described by (11) with the parameters ρ.

To compare the model performance subject to the parameters ρ we use a loss function $\pi(u) : \mathbb{R} \to \mathbb{R}$, which has the following properties:

- $\pi(0) = 0$,
- $\pi(u) \geqslant 0$ for all $u \in \mathbb{R}$,
- $\pi(u)$ is non-increasing in $u < 0$, and non-decreasing in $u > 0$.

The parameter identification for both the mean and variance using the model (11) is a particular case of the M-estimation one [33].

The identification problem for the mean, described by the model (11) is to find

$$\rho^* \in \operatorname*{Argmin}_{\substack{\rho \in \mathbb{R}^{3N+M+2}: \\ i)-iv)}} \sum_{r=1}^{R} w_r \pi(\Delta_r^{\mathcal{M}}(\rho)). \tag{16}$$

The identification problem for the variance, described by the model (11) looks similar and is to find

$$\rho^* \in \operatorname*{Argmin}_{\substack{\rho \in \mathbb{R}^{3N+M+2}: \\ i),\, iv)}} \sum_{r=1}^{R} w_r \pi(\Delta_r^{\mathcal{D}}(\rho)). \tag{17}$$

The stress testing and subsequent data processing for the model (11) assume the following steps.

1. For each pair (x^r, y^r) one executes T independent tasks (here, T is a sample size), and registers the runtimes $\{s_t^r\}_{t=\overline{1,T}}$.
2. The set $\{\widetilde{\mathcal{M}}^r, \widetilde{\mathcal{D}}^r\}_{r=\overline{1,R}}$ is calculated by the well-known formulae

$$\widetilde{\mathcal{M}}^r = \frac{1}{N} \sum_{t=1}^{T} s_t^r, \qquad \widetilde{\mathcal{D}}^r = \frac{1}{N-1} \sum_{t=1}^{T} \left(s_t^r - \widetilde{\mathcal{M}}^r \right)^2.$$

Further, we formulate the parameter identification problem for the model (14). Let $\{(x^r, y^r)\}_{r=\overline{1,R}}$ be a fixed collection of the variables (x, y), chosen for the stress testing. Its result is processed into the set $\{\widetilde{\mathcal{M}}^r, \widetilde{\mathcal{D}}^r\}_{r=\overline{1,R}}$ of the sample moments, and the vector $\widetilde{z}$, estimating the node state parameters. Let also the set $\{w_r\}_{r=\overline{1,R}}$ of positive observation weights be given. The value

$$\delta_r^{\mathcal{M}}(\rho) \triangleq \widetilde{\mathcal{M}}^r - h_{\widetilde{z}}(x^r, y^r, \rho)$$

is the residual of the observed mean $\widetilde{\mathcal{M}}^r$ corresponding to the triplet $(x^r, y^r, \widetilde{z})$, described by the model (14) with the parameters ρ. Similarly, the value

$$\delta_r^{\mathcal{D}}(\rho) \triangleq \widetilde{\mathcal{D}}^r - h_{\widetilde{z}}(x^r, y^r, \rho)$$

is the residual of the observed variance $\widetilde{\mathcal{D}}^r$ corresponding to the triplet $(x^r, y^r, \widetilde{z})$, described by the model (14) with the parameters ρ.

The identification problem for the mean, described by the model (14) is to find

$$\rho^* \in \operatorname*{Argmin}_{\substack{\rho \in \mathbb{R}^{J(3N+M+2)}: \\ i)-v)}} \sum_{r=1}^{R} w_r \pi(\delta_r^{\mathcal{M}}(\rho)). \tag{18}$$

The identification problem for the variance, described by the model (14), looks similar and is to find

$$\rho^* \in \operatorname*{Argmin}_{\substack{\rho \in \mathbb{R}^{J(3N+M+2)}: \\ i),\, iv)}} \sum_{r=1}^{R} w_r \pi(\delta_r^{\mathcal{D}}(\rho)). \tag{19}$$

The stress testing and subsequent data processing for model (14) assume the following steps.

1. For each pair (x^r, y^r) one executes T independent tasks (here, T is a sample size) and registers the execution times $\{s_t^r\}_{t=\overline{1,T}}$. In the process of each task execution, one also registers the auxiliary data $\{\zeta_t^r\}_{\substack{t=\overline{1,T}\\ r=\overline{1,R}}}$ which helps to recover the node state parameters z.
2. One calculates the sample moments $\{\widetilde{\mathcal{M}}^r, \widetilde{\mathcal{D}}^r\}_{r=\overline{1,R}}$.
3. Using the collected data $\{s_t^r\}_{\substack{t=\overline{1,T}\\ r=\overline{1,R}}}$ one obtains the exact value of the node state parameters, or some estimate $\widetilde{z}$.

Let us remind the reader that the parameters z are responsible for the partition $\{\mathcal{U}_j(z)\}_{j=\overline{1,J}}$ of the whole set $\mathcal{U}$.

The structure of the data $\{\zeta_t^r\}_{\substack{t=\overline{1,T}\\ r=\overline{1,R}}}$ and the procedure of the collection are individual for each specific user task.

Usually, it is sufficient to periodically register the following parameters:

– percentage of the central processing unit (CPU) usage,
– the volume of RAM used,
– the volume of data read,
– the volume of data written,
– the total of swap memory used,
– the volume of swap read,
– the volume of swap written, etc.

In some cases, necessary information can be obtained by various diagnostic tools provided by the OS, for example, System Monitor in Microsoft Windows.

To illustrate the procedure of the vector z restoration or estimation and its utilization for the definition of the partition family $\{\mathcal{U}_j(z)\}_{j=\overline{1,J}}$, we propose two simple examples.

The first example is devoted to the rendering of a 3D scene. There is only one resource parameter x_1: it defines an amount of the node RAM. Each user task is also defined by the single parameter y_1, which is the size of the file under processing. The current node state is defined by the scalar z_1, a RAM volume available for application usage. The application attempts to load the whole 3D-scene file into the RAM and perform its visualization. If it is conducted, the processing procedure is quick, otherwise, the processing time increases significantly because of the swapping mechanism. In this simple situation, the initial set $\mathcal{U}$ is partitioned into two subsets:

$$\mathcal{U}_1(z_1) = \{(x_1, y_1) \in \mathcal{U} :\ z_1 - y_1 \geqslant 0\}$$

which corresponds to the case of the swapping absence, and

$$\mathcal{U}_2(z_1) = \{(x_1, y_1) \in \mathcal{U} :\ z_1 - y_1 < 0\}$$

which corresponds to the swapping. The value z_1 can be obtained precisely by diagnostic applications like Windows Task Manager (in the performance tab) or the function *virtual_memory()* in the PSUTIL Python cross-platform library.

We emphasize that this is just a simple clarifying example: indeed, the RAM volume available for the application purposes is not constant and depends on the total RAM size. In addition, there are various factors that can reduce the available RAM size, such as the startup of OS housekeeping processes, the growth in the size of non-paged areas as the total amount of RAM, or the number of processor cores used increases.

The second example presents some information processing in a database. The local distinctions of the task runtime lead to piece-wise model (14), which are also caused by the swapping mechanism.

Each user task is defined by the single parameter y_1, which is the number of database records under processing. Again, the parameter z_1 represents the RAM volume available for the application usage. The dependency of the used RAM volume ζ_1 on the task parameter y_1 is an affine one

$$\zeta_1(y_1) = z_2 + z_3 y_1.$$

Here, z_2 is a parameter defining a constant RAM expense during the task execution, and z_3 is the RAM unit expense per one processed record. One can collect the statistics ζ_1, using the function *virtual_memory()*, and calculate the estimates $\widetilde{z}_2$ and $\widetilde{z}_3$ as the parameters of the linear regression. Finally, in the example, the initial set $\mathcal{U}$ is partitioned into two subsets:

$$\mathcal{U}_1(z_1, z_2, z_3) = \{(x_1, y_1) \in \mathcal{U} :\ z_1 - z_2 - z_3 y_1 \geqslant 0\}$$

in the case of no swapping, and

$$\mathcal{U}_2(z_1, z_2, z_3) = \{(x_1, y_1) \in \mathcal{U} :\ z_1 - z_2 - z_3 y_1 < 0\}$$

otherwise.

Again, we consider here a simplified situation. Indeed, the RAM volume available for application purposes is not constant and depends on the whole RAM size and the number of processing cores. The parameter z_2 may also depend on the node resources.

The section below illustrates the usage of the proposed time complexity surfaces.

5. Numerical Study of Time Complexity Surfaces for Personal Data Anonymization System

5.1. Synopsis of Anonymization System Prototype

To illustrate the applicability of the proposed stochastic time complexity surfaces describing task running time, we choose an application that is a prototype of the passengers' personal data anonymization system [31]. The virtual computing node is deployed on the host computer with the following characteristics: Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.1 GHz, 64 Gb RAM, 256 Gb SSD, Windows Server 2016 Standard operating system, Microsoft Hyper-V hypervisor. The guest software of the node consists of

– the system software: Windows Server 2012 R2 Standard operating system, Microsoft(R) SQL Server(R) 2012 relational database management system,
– the application software, including a database with approximately 26 million records and program components, realizing the graphic interface and anonymization templates created in the Microsoft Visual Studio 2017 development environment.

The vector of the node resources $x = (x_1, x_2)$ includes the following components:

– x_1, which is the number of processing cores (up to 16),
– x_2, which is a RAM volume (up to 16 Gb).

The tested anonymization software provides the fulfillment of the functions below:

1. the choice of the personal data bulk by the filter system for the subsequent processing,
2. anonymization of the chosen data by the various templates: exclusion, grouping, random permutation, noise adding, and the artificial data synthesis,
3. calculation of the anonymity level for the chosen combination of the fields,
4. calculation of the sample correlation of the chosen fields,
5. utility level calculation for the chosen combination of the fields,
6. histogram drawing for the chosen numerical field.

In spite of the possible variety of user tasks, which could be solved by the application software, we investigate two types of tasks:

1. combination of functions 1 and 2, i.e., choosing the personal data bulk with subsequent anonymization of all "sensetive" fields in the records; the task is completely characterized by the scalar y_1, which is the number of personal data records;
2. combination of the functions 1 and 4, i.e., choosing the personal data bulk and the pair of numerical fields among them with subsequent calculation of some statistics; the task is completely characterized by the scalar y_1, which is the sample size.

First, a user applies function 1 in most cases. Indeed, he/she processes only some subset of the personal data extracted by the filters. Second, from the algorithmic point of view, one can separate functions 2–6 into two subsets. The first one contains functions 2 and 5, realized by some sequential "linear" algorithms. The second subset includes functions 3, 4, and 6, realized by the standard optimized database procedures. For a demonstration of the high performance of the stochastic time complexity surfaces approach in practice, we choose one function from each subset.

To automate the data selection for the subsequent processing in the stress testing, we choose a range of personal data by its offset and length for subsequent processing (all records are numbered in ascending order starting from 1).

5.2. Time Complexity Surfaces for Anonymization Procedure

The anonymization procedure represents the selection of the initial personal data from the database and its further transformation by the anonymization templates by saving the results in the database. All records are processed in the RAM, which results in considerable strain on the RAM volume. The templates are applied sequentially. That is why the additional CPU cores do not give a time advantage.

The initial data contains

- the primary key integer field "Serial number",
- the integer field "Age",
- the date/time fields "Start Date" and "Finish Date",
- the real fields "Result 1"–"Result 3",
- categorical fields "Category 1"–"Category 4", "Airport" and "Company".

There is an allotment of the anonymization templates applied to the fields above:

- the exclusion template— to the field "Serial number",
- the grouping template—to the field "Age",
- the noise adding template—to the combination of the fields "Start Date", "Finish Date", "Result 1"–"Result 3", "Category 1"–"Category 4",
- the synthesis template—to the combination of the fields "Airport" and "Company".

We fulfill the stress testing to collect the statistical data for the parameter identification. The experiment plan is defined by the set of triplets $\{(x_1^r, x_2^r, y_1^r)\}$. Here, the first parameter x_1—the number of the processor cores—varies from 1 to 16 by doubling. The second parameter x_2—the RAM volume—varies from 2 to 16 Gb with steps of 2 Gb. The third parameter y_1—the number of the processed records—varies in the following way: 10,000, 50,000, and further with steps of 50,000 up to 500,000. To calculate the sample mean and variance, we repeat each experiment 20 times.

Before the long process of stress testing, we perform auxiliary statistical study to compare the influence of the soft/hardware functioning fluctuations with one of the datasets chosen for the processing. We configure a virtual node with 8 Gb RAM and two processor cores to anonymize 50,000 personal data records. We produce 1000 experiments with these tasks. They differ from each other only by the chosen processed data:

- Each of 1000 tasks processes the same set of the 50,000 sequential personal database records with fixed offset #1,
- Each of 1000 tasks processes the same set of the 50,000 sequential personal database records with fixed offset #2,
- Each of 1000 tasks processes 50,000 sequential personal database records with a random uniformly distributed offset.

Figure 1 contains Parzen estimates of the runtime probability density function (pdf), corresponding to the experiments outlined above.

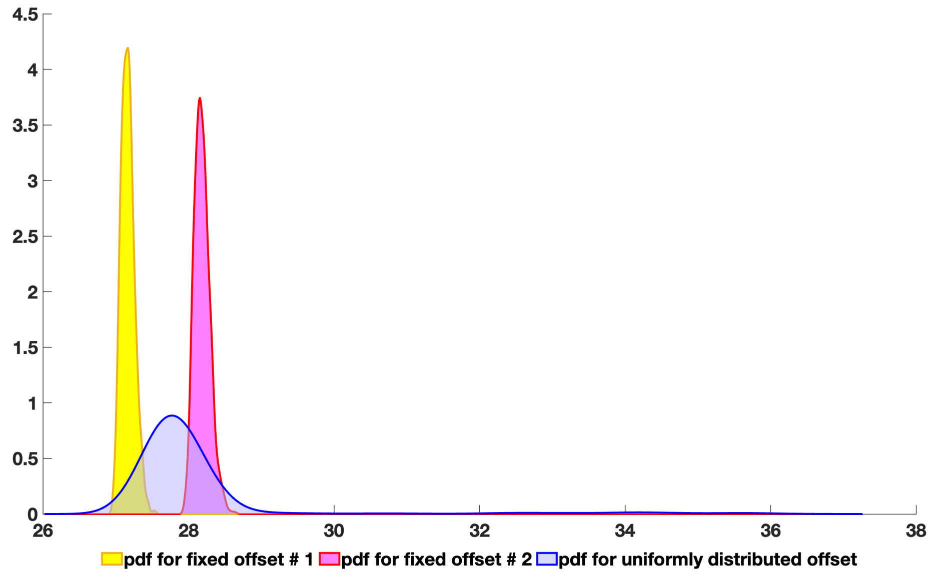

Figure 1. Anonymization procedure: Parzen estimates of time pdf for various processed data.

One can see that the range of the sample corresponding to the fixed offsets # 1 and 2 constitute just 6–8% of the sample range with the uniformly distributed offset. So, the major source of the statistical uncertainty in the task execution time is formed by the processed data choice. Note that in the case of the uniformly distributed offset the pdf estimate has a fat right tail.

During the stress testing, we register some auxiliary statistical data (see Section 4) containing indirect information concerning the stable operation of each specific node configuration. In particular, we log the data volume read/written in the swap memory during the user task execution. This data is drawn in Figure 2.

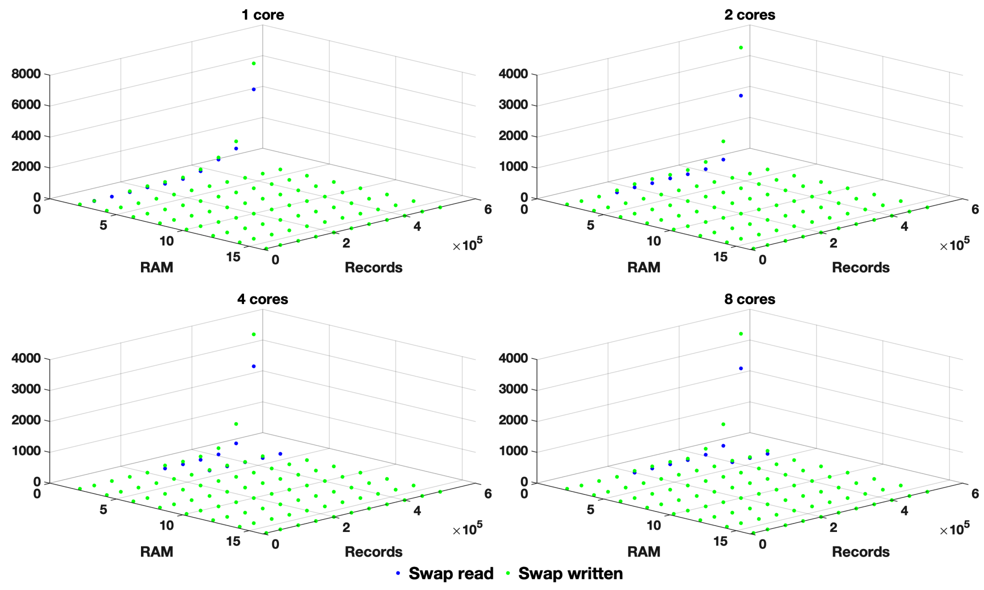

Figure 2. Anonymization procedure: volumes of swap read and written.

From the figure, one can conclude that the configurations with 2 Gb RAM are prone to intensive swap read/write operations. Evidently, swapping as a routine part of the user task processing can not be considered as an effective technique, realizing the anonymization procedure. Hence, we exclude all "poor" node configurations with 2 Gb RAM from the area of the model parameter identification.

Figures 3 and 4 contain the available sample mean and variance, drawn by color bullets for one, two, four and eight processor cores. Here and below in the paper, the case of 16 cores is omitted in the figures because it looks similar to the case of eight cores.

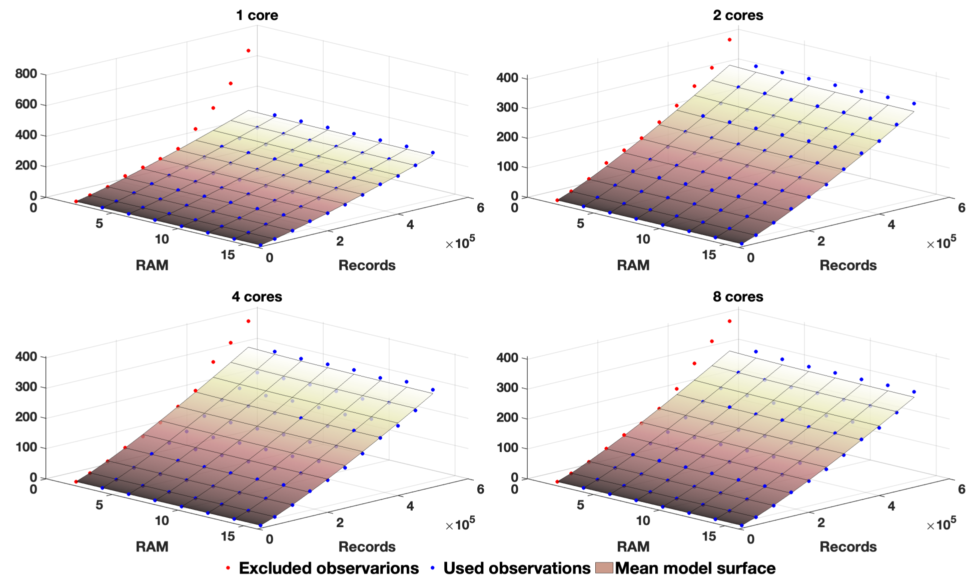

Figure 3. Anonymization procedure: stress testing results and mean model surface.

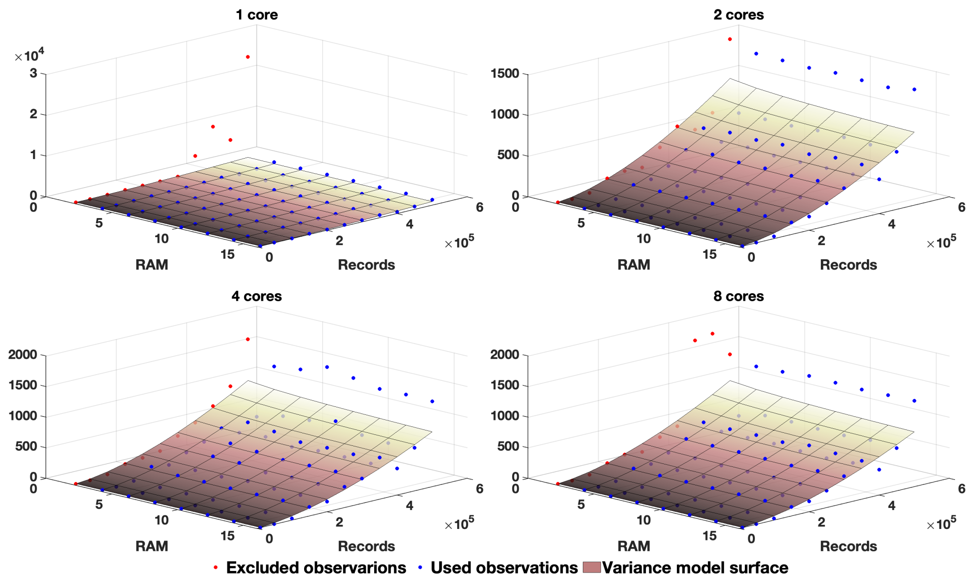

Figure 4. Anonymization procedure: stress testing results and variance model surface.

Note that the excluded configurations with 2 Gb RAM demonstrate an inappropriately high mean execution time. The data chosen for the model identification look smooth, and we use the simple model $g(\cdot)$ (11) to describe both the mean and variance.

To provide robustness for the estimates of the model parameters, we choose the error absolute value as the loss function $\pi(u) = |u|$. The obtained stochastic time complexity surfaces for the mean and variance and the corresponding coefficients of determination have the form:

$$\mathcal{M}(x_1, x_2, y_1) = 5.700 + 6.880 \times 10^{-5} \times x_1^{-0.043} x_2^{-0.014} y_1^{1.174}, \qquad R^2_{\mathcal{M}} = 0.995,$$

and

$$\mathcal{D}(x_1, x_2, y_1) = 3.470 \times 10^{-8} \times x_1^{0.001} x_2^{-0.062} y_1^{1.625}, \qquad R^2_{\mathcal{D}} = 0.827.$$

For the visual comparative analysis, Figures 3 and 4 also contain the proposed models. One can draw the following conclusions. First, the visual similarity of the available observations and the calculated surfaces confirms the high performance of the proposed model for this type of user task. The high determination coefficient also approves this deduction.

Second, the choice of smooth simple model g (11) is reasonable: in this case, it is sufficient for the estimation with the appropriate quality.

Third, the proposed sketch of the anonymization algorithm as a one-thread consecutive process seems true. As the available RAM volume is enough to allocate all processed data, the further RAM extension and the rise in the number of cores does not affect the moment characteristics of the task runtime. The small powers of the variables x_1 and x_2 represent numerical confirmation of this fact.

Fourth, the proposed model (11) is redundant in this case: some of the parameters equal zero: $d_1 = d_2 = 0$ for the mean time, and $a = d_1 = d_2 = 0$ for the variance surface. This is one more confirmation that after the RAM volume is enough to allocate all the processed records, the further RAM extension does not move the mean and variance to smaller or greater values.

5.3. Time Complexity Surfaces for Statistical Queries

The statistical query under testing represents the selection of the initial personal data records from the database with subsequent calculation of the sample covariance of the chosen fields. That is mainly standard procedure, fulfilled by the optimized database routines. One can expect high parallelism, so enlargement of the processor core number could give an advantage in the task running time. Minimal time is attained when all processed data is located in the file cache of the OS, so the RAM volume plays a key role again.

In the investigated case, we calculate the sample covariance between the fields "Start Date" and "Finish Date" of the initial personal data. The field values belong to the "Date-Time" type, so before the processing, they are converted into real numbers.

Again, before the stress testing, we fulfill auxiliary statistical research to compare the influence of the soft/hardware functioning fluctuations with one of the data chosen for the processing. We use a virtual node with 8 Gb RAM and two processor cores to calculate the covariance by the sample of size 12,500,000. We produce 1000 experiments with these tasks. We choose the processed data by analogy with Section 5.2. Figure 5 contains Parzen estimates of the task execution time pdf, corresponding to the abovementioned experiments.

By contrast with the anonymization procedure, the range of the sample corresponding to the fixed offsets # 1 and 2 constitute just 14–24% of the sample range with the uniformly distributed offset. All the pdf estimates demonstrate the polymodal character, and have no fat tails.

To collect the statistical data for the parameter identification we perform stress testing. The experiments are defined by the triplets $\{(x_1^r, x_2^r, y_1^r)\}$. Here, x_1 is a number of processor cores, and x_2 is the RAM volume. The set of virtual node configurations for the statistical query stress testing coincides with the one for the anonymization procedure. The third parameter—y_1—represents the sample size and varies from 2,000,000 to 20,000,000 with steps of 2,000,000. To calculate the sample mean and variance, we repeat each experiment 50 times.

During the stress testing, we register the auxiliary statistical data, in particular, the volume of the used swap memory during the user task execution. The collected observations are presented in Figure 6.

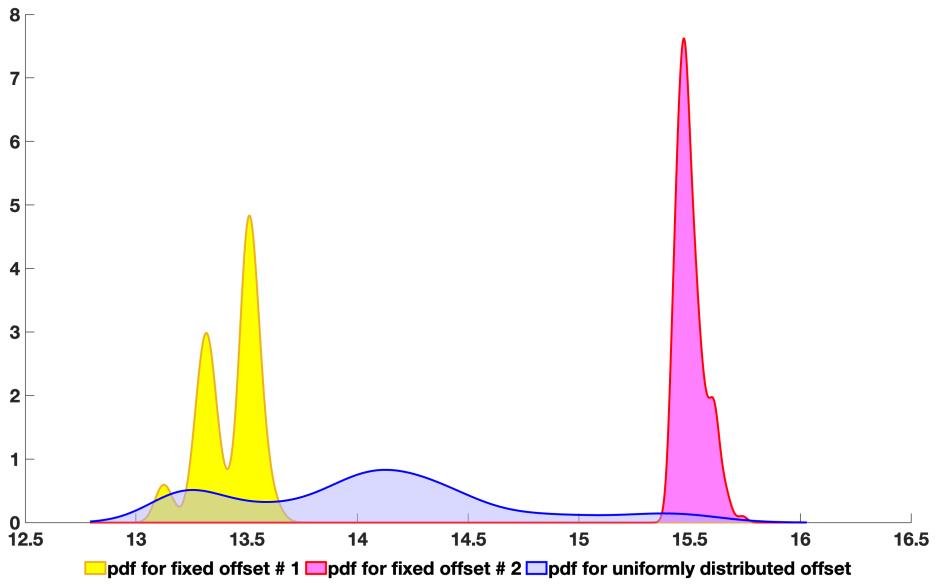

Figure 5. Statistical queries: Parzen estimates of time pdf for various processed data.

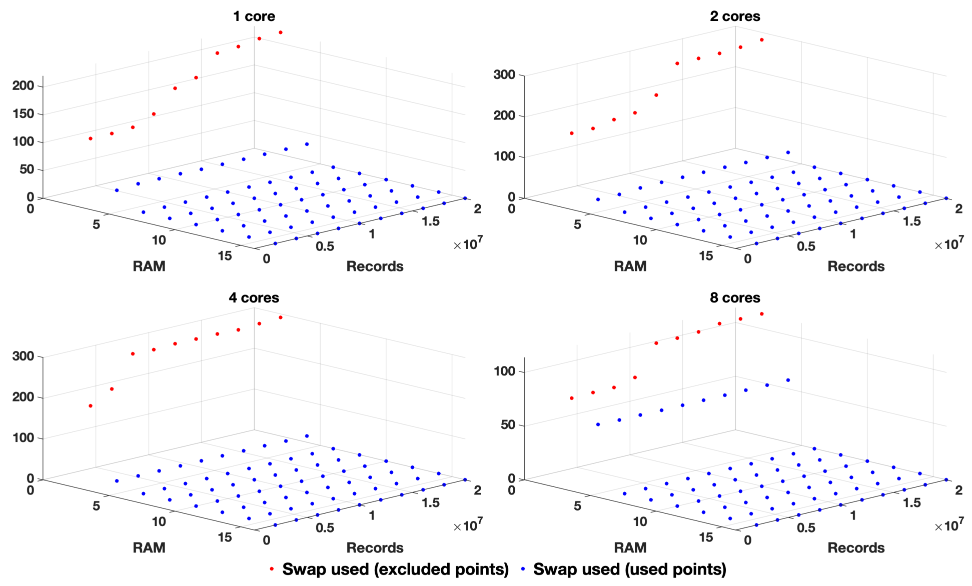

Figure 6. Statistical queries: volumes of swap used.

We can see that the configurations with 2 Gb RAM are accompanied by the swap usage. By contrast with the anonymization procedure, these are not the read/write processes but the static occupation of some part in the swap file. Quite possibly, this is the data that the OS does not need to run the testing software. We consider this technique as a compulsory measure and assume that similar configurations should be avoided. Hence, we exclude all "poor" node configurations with 2 Gb RAM from the model parameter identification.

Figures 7 and 8 contain the available sample mean and variance along with the calculated complexity surfaces. The observations, corresponding to the excluded configurations with 2Gb RAM, demonstrate an inappropriately high mean runtime.

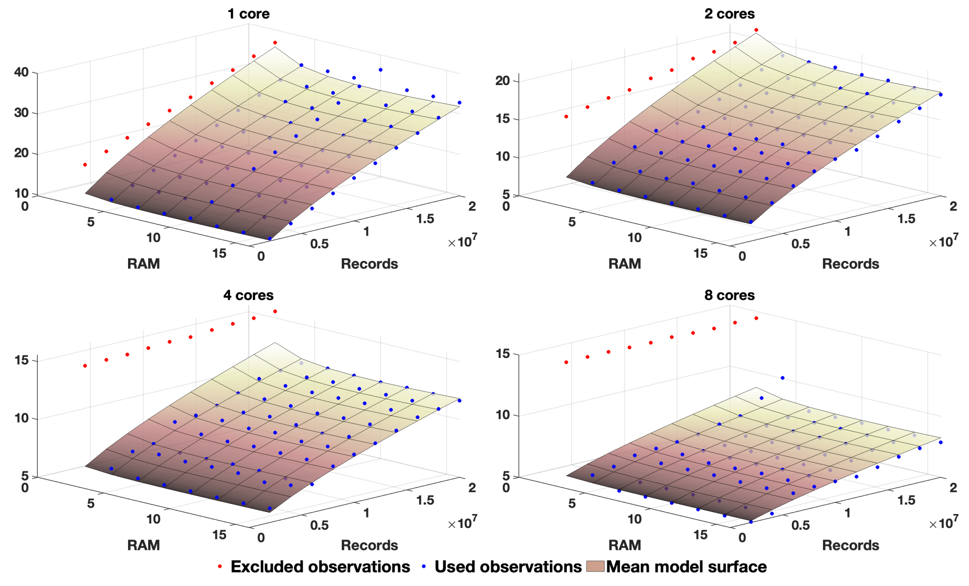

Figure 7. Statistical queries: stress testing results and mean model surface.

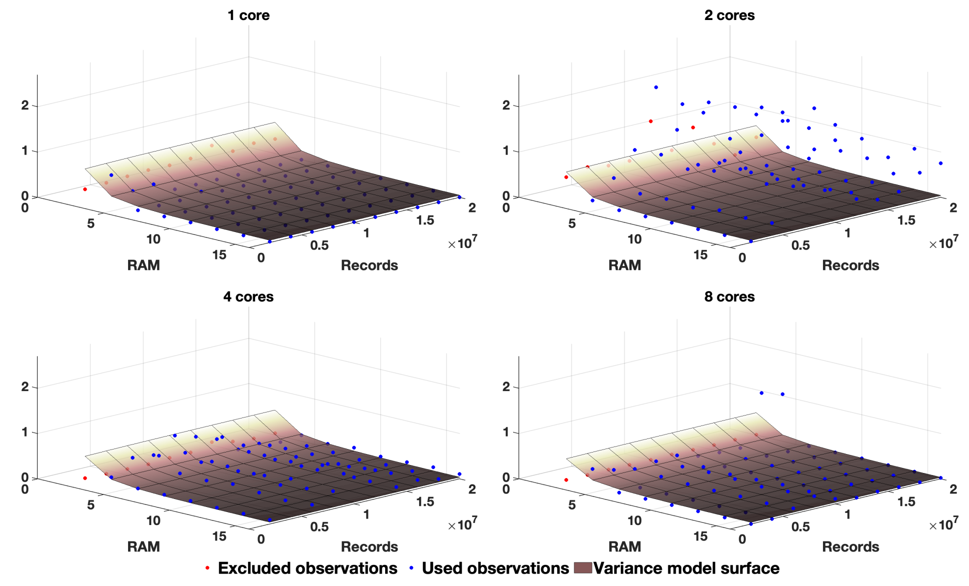

Figure 8. Statistical queries: stress testing results and variance model surface.

Again, we choose the error absolute value as the loss function $\pi(u) = |u|$, and the simple model $g(\cdot)$ (11). The obtained stochastic time complexity surfaces and the corresponding coefficients of determination have the form:

$$\mathcal{M}(x_1, x_2, y_1) = 4.4346 + 2.767 \times 10^{-4} \times x_1^{-0.941} x_2^{-0.272} (1 + 0.061 x_2^{0.89}) y_1^{0.697}, \quad R^2_{\mathcal{M}} = 0.991, \tag{20}$$

and

$$\mathcal{D}(x_1, x_2, y_1) = 1.100 \times 10^{-7} + 1.786 \times x_1^{-0.163} x_2^{-2.0} y_1^{0.023}, \qquad R^2_{\mathcal{D}} = -0.158. \tag{21}$$

One can draw the following conclusions. First, the comparison of the available observations and constructed stochastic time complexity surface demonstrates the quality of the proposed model for the mean time. The high determination coefficient also confirms this deduction. Visually, model (11) describes the variance moderately. The negative determination coefficient does not contradict logic: the estimate of the least absolute values can have an error with the $\mathcal{L}_2$-norm greater than the sample mean. Briefly, the sample variance observations oscillate actively, and the oscillation intensity is maximal for two processor cores.

Second, the assumption concerning the optimization of the standard statistical calculations by Microsoft SQL Server and their high parallelization seems true: in the mean time model, the power of the variable x_1 equals -0.941, i.e., the mean value is almost in inverse proportion with the number of processor cores.

Third, after some volume, the further RAM extension does not lead to the mean task running time and, moreover, it demonstrates weak growth.

5.4. Inferences and Recommendations after Stress Testing of Anonymization System

The stress testing with the subsequent design of the stochastic time complexity surfaces is not a final goal in itself. The obtained model is an informational support for the planning of the virtual computing nodes and a tool for solutions to Problems 1 and 2, which are real for the provider. As an illustrative example, we demonstrate the model possibilities in application to the tested personal data anonymization system.

First, the provider should avoid the node configurations with a RAM volume of less than or equal to 2 Gb to provide a relatively stable node operating environment while it processes a user task. Second, to solve Problems 1 and 2, the provider has to use Formulas (5) and (6): the time distributions of both task types are neither unimodal nor concave. Third, the proposed conservative estimates are reasonable in practice for a small ratio $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}}$, not more than, say, 0.2. This small ratio guarantees for Problem 2 that the conservative threshold $\overline{T}$ is comparable with the mean time and does not exceed it by orders of magnitude.

Figure 9 demonstrates the surfaces of this ratio, calculated for the anonymization procedure. One can see that it does not exceed 0.18.

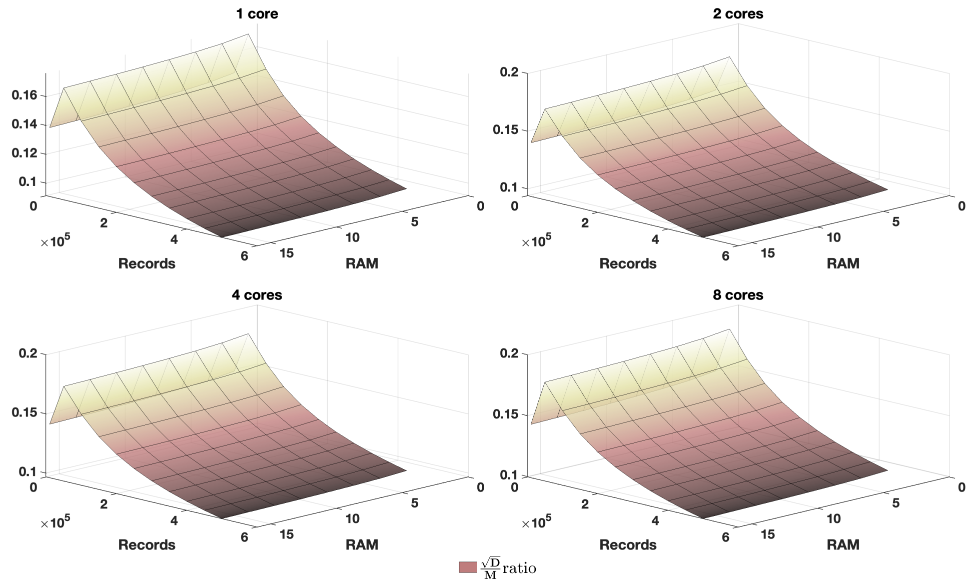

Figure 9. Anonymization procedure: $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}}$-ratio.

Figure 10 contains the solution to Problem 2 for the data anonymization procedure of 500,000 records: dependency of the threshold value $\overline{T}$ on the cores number (1, 2, 4, 8, 16) given the fixed RAM volume 4, 8 and 16 Gb. The confidence level is 0.01. We include in the figure the model values of the mean task execution time $\mathcal{M}$, provided by the identified model.

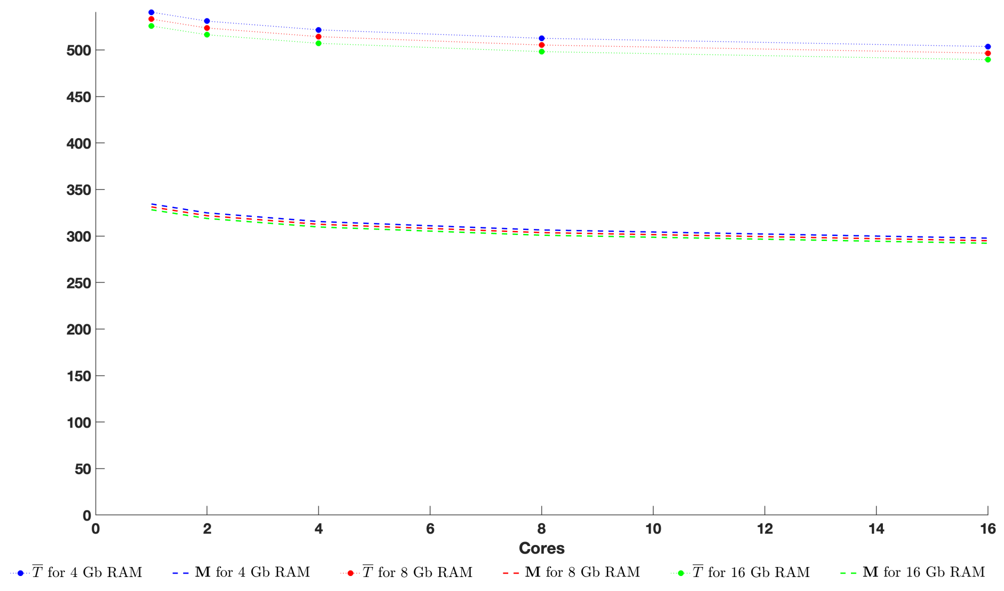

Figure 10. Anonymization of 500,000 records: solution $\overline{T}$ to Problem 2.

Comparing the corresponding values, we can conclude that $\overline{T}$ are more than $\mathcal{M}$ at most at 70%: this gap looks significant but not excessive. Also, we can see that neither RAM extension nor the growth of the CPU core number leads to the reduction in the threshold $\overline{T}$. From the provider's point of view, if the user task represents anonymization of at most 500,000 records, then it is economically reasonable to equip the virtual computing node with the minimal number of processor cores (say, 1–2) and minimal RAM volume (at least 4 Gb), and set SLA threshold to be equal to 550 s.

Figure 11 demonstrates the surfaces of the ratio $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}}$, calculated for the statistical query calculation. One can see that it does not exceed 0.12.

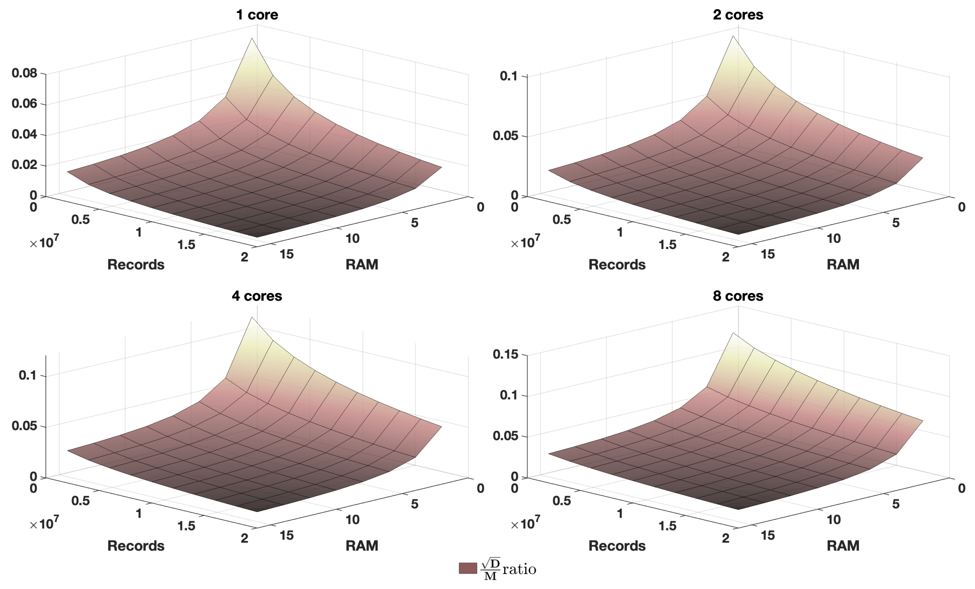

Figure 11. Statistical queries: $\frac{\sqrt{\mathbf{D}}}{\mathbf{M}}$-ratio.

Figure 12 contains the solution to Problem 2 for the statistical query calculation over the sample of 20,000,000 records: dependency of the threshold value $\overline{T}$ on the core number (1, 2, 4, 8, 16) given the fixed RAM volume 4, 8 and 16 Gb. Again, the confidence level is 0.01. We include in the figure the values of the mean task running time $\mathcal{M}$, provided by the identified model. Comparing the corresponding values, we can conclude that the values of $\overline{T}$ are more than $\mathcal{M}$ at most at 10%.

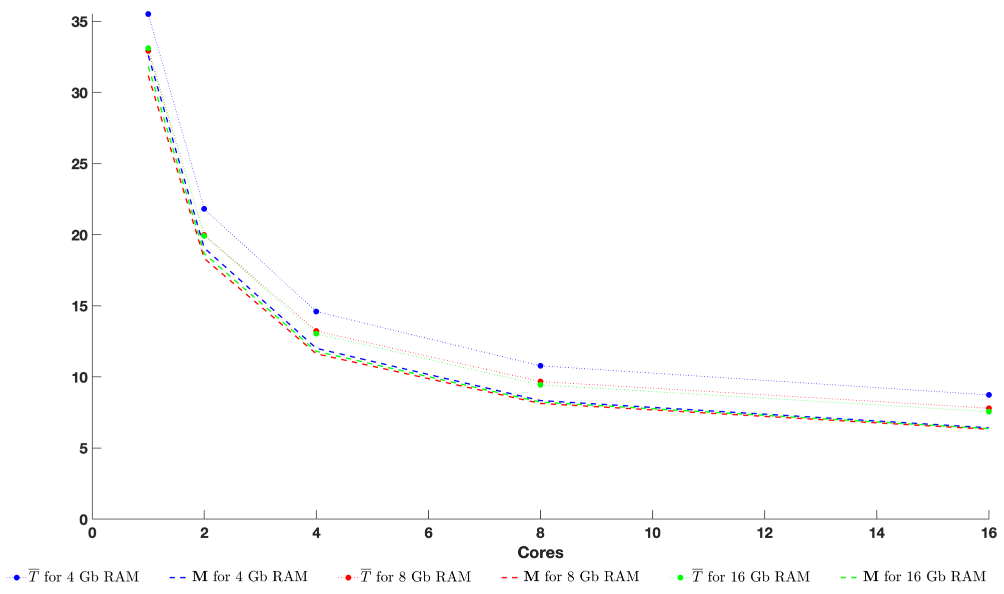

Figure 12. Statistical query for 20,000,000 records: solution $\overline{T}$ to Problem 2.

By analogy with the case of the anonymization procedure, the RAM extension does not affect the value $\overline{T}$. By contrast, the growth of the processor cores number is almost inversely proportional to $\overline{T}$. So, if the user task represents the calculation of the statistical characteristics given the sample of the length no more than 20,000,000, then it is economically reasonable to equip the virtual node with four processor cores and minimal RAM volume (at least 4 Gb) and set the SLA threshold $\overline{T}$ to be equal to 15 s. The growth of the core number from one until four facilitates the reduction of $\overline{T}$ by more than twice.

To verify the performance of the proposed model and its applicability to the solution for Problems 1 and 2 once more, we fulfill additional experiments with the statistical queries:

– the node with two cores and 8 Gb RAM is used for the calculation of the covariance given the sample of the size 12,500,000; we repeat this test 1000 times,
– the node with two cores and 4 Gb RAM is used for the calculation of the covariance given the sample of the size 20,000,000, we repeat this test 500 times.

That is quite a new statistic, and we do not use it in the model identification procedure. The processed data is chosen randomly and independently: in each experiment, the offset is uniformly distributed. For both configurations, we solve Problem 2, i.e., calculate the threshold $\overline{T}$ with the confidence level 0.01, using models (20) and (21). For the experiments, we are interested in the number of outcomes, which exceed the corresponding thresholds $\overline{T}$.

Table 2 presents the experiment results: for both configuration it contains

– the mean $\widetilde{\mathcal{M}}$ calculated by the tested sample,
– the variance $\widetilde{\mathcal{D}}$ calculated by the tested sample,
– the sample minimum $\underline{\mathfrak{T}}$,
– the sample maximum $\overline{\mathfrak{T}}$,
– the mean $\mathcal{M}$, calculated by model (20),
– the variance $\mathcal{D}$, calculated by model (21),
– the solution $\overline{T}$ to Problem 2, calculated by models (20) and (21),
– the number N of outcomes, which exceed $\overline{T}$.

Table 2. Results of validation tests.

	$\widetilde{\boldsymbol{\mathcal{M}}}$	$\widetilde{\boldsymbol{\mathcal{D}}}$	$\underline{\boldsymbol{\mathfrak{T}}}$	$\overline{\boldsymbol{\mathfrak{T}}}$	$\boldsymbol{\mathcal{M}}$	$\boldsymbol{\mathcal{D}}$	$\overline{\boldsymbol{T}}$	**N**
Configuration 1	14.056	0.730	13.071	15.713	14.484	0.036	15.739	0
Configuration 2	18.582	0.999	17.317	25.403	19.104	0.147	21.629	2

The provided tests seem challenging. First, all configurations with two cores demonstrate a high oscillating sample variance. Second, the sample size 12,500,000 is not used in the stress testing, this is an "intermediate" value. Third, 20,000,000 is the maximal sample size in the stress testing process.

As is expected, model (21) underestimates the variance, comparing with the corresponding sample values. Nevertheless, the guaranteeing threshold $\overline{T}$ plays its role perfectly: none of the 1000 observations corresponding to the first configuration exceed $\overline{T}$. In the second configuration, only two observations out of 500 exceed $\overline{T}$.

6. Conclusions

The running time of a user task is very complicated to formally describe, particularly in terms of probability theory. The polymodality of the distribution, fat tails, and dependence on the unobservable current node state complicate the development of the mathematical models for the solution to the collection of applied providers' problems. From this point of view, the proposed stochastic time complexity surfaces look rather efficient. For the solution to Problems 1 and 2, the model is better than the well-known average-case complexity one because the obtained guaranteeing thresholds are less conservative. Furthermore, the set of stress testing experiments for the identification of the average-case complexity model coincides with the one for the stochastic time complexity surfaces, but one has to register an extended set of statistical data. The offered families of the functions, g, (11), h, (14), look acceptable to describe these surfaces. In general, the runtime $\tau(\omega, x, y)$ represents the whole family of the random values, parameterized by the amount of the node resource variables x and user task ones y. Hence, the mean value $\mathcal{M}$ and variance $\mathcal{D}$ of the runtime have to be functions of the pair (x, y), and these functions are just the time complexity surfaces. The description of the runtime by the proposed model is equivalent to the one of a random value by its mean and variance. The stochastic surfaces model allows us to solve the same class of applied problems as the knowledge of two first moments of a random value.

The current version of the suggested time complexity model and the parameter identification methodology seems quite applicable for the solution to the provider's planning problems. However, there are several heterogeneous limitations to the proposed approach. The reasons for this are the theoretical and numerical complexity of the parameter identification problem and the high resource and time costs for the stress testing process. All the issues seem transparent, and below we focus on their solutions, indicating the prospective areas of investigation that can enhance the model, the performance of its parameter identification, and stress testing efficiency.

1. *The improvement of stress test planning.* Even during the stress testing of "the toy" anonymization system, it becomes clear that the testing is a very costly process that should be optimized. The automatic extensive growth of the plan points (x^r, y^r) along with the sample size T is not the correct way to increase the identification performance. In fact, the corresponding formal justified assertions in the experiment planning are in demand. The assertions should characterize the estimation accuracy and consistency and indicate the ways to reduce the test experiment volume. Furthermore, the provider should use the available a priori information about the distribution in the experiment planning. As is shown in the applied part of the paper, this distribution is a crucial factor that affects the model parameters. Finally, one should develop adaptive

versions of the identification procedure to use novel statistical data registered during the routine operation of the node.

2. *The improvement of the raw data preprocessing and the parameter identification procedures.* It is important to understand the technical details to formally describe the interconnection between the task running time and the current node state. It could help to explain the polymodality of the time distribution: clustering of the runtime can be caused by the parallel activities of some OS services like backup or software updates. Further, the key tools for the solution to the identification problems (16)–(19) are the instruments of numerical optimization. The mentioned problems are truly challenging because they belong to the set of non-smooth and non-convex constrained optimization. To solve the illustrative problem in the paper, we use the routines from the standard MATLAB R2019a optimization kit. Evidently, the solution to the identification problem and the online parameter adaptation ones for the real computing nodes requires the utilization of advanced optimization techniques, including classical algorithms [34] and contemporary metaheuristic ones [35].
3. *The enhancement of the surface for the runtime variance.* It is possible to extend the class of the admissible model functions. Another way is to change the point of view towards the variance estimation. Indeed, the provider does not need an accurate variance model, but some conservative estimate. He/she can obtain it by replacement of the $\mathcal{L}_1$ loss function by some piece-wise linear one, usually used in the quantile regression [36].

These topics, as mentioned, represent promising avenues for future research.

Author Contributions: Conceptualization, A.B.; methodology, A.B.; software, A.I.; validation, A.B. and A.I.; formal analysis, A.B. and A.I.; investigation, A.B.; resources, A.I.; data curation, A.I.; writing, A.B. and A.I.; visualization, A.B.; supervision, A.B.; funding acquisition, A.B. All authors have read and agreed to the published version of the manuscript.

Funding: The research has been supported by the Ministry of Science and Higher Education of the Russian Federation, project No. 075-15-2020-799.

Data Availability Statement: Data sharing is not applicable to this article.

Acknowledgments: The research was carried out using the infrastructure of the Shared Research Facilities “High Performance Computing and Big Data” (CKP “Informatics”) of the Federal Research Center “Computer Science and Control” of the Russian Academy of Sciences.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CE	Conditional Expectation
CPU	Central Processing Unit
HMM	Hidden Markov Model
OS	Operation System
pdf	Probability Density Function
QoS	Quality of Service
RAM	Random Access Memory
SLA	Service Level Agreement

References

1. Sakellari, G.; Loukas, G. A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. *Simul. Model. Pract. Theory* **2013**, *39*, 92–103. [CrossRef]
2. Jawed, M.S.; Sajid, M. A Comprehensive Survey on Cloud Computing: Architecture, Tools, Technologies, and Open Issues. *Int. J. Cloud Appl. Comput.* **2022**, *12*, 1–33. [CrossRef]

3. Vijayakumar, P.; Rajalingam, P.; Rajeswari, S.V.K.R. Edge Computing Optimization Using Mathematical Modeling, Deep Learning Models, and Evolutionary Algorithms. In *Simulation and Analysis of Mathematical Methods in Real–Time Engineering Applications*; Kumar, T.A., Julie, E.G., Robinson, Y.H., Jaisakthi, S.M., Eds.; Wiley: Hoboken, NJ, USA, 2021; Chapter 2, pp. 17–44.
4. Saeik, F.; Avgeris, M.; Spatharakis, D.; Santi, N.; Dechouniotis, D.; Violos, J.; Leivadeas, A.; Athanasopoulos, N.; Mitton, N.; Papavassiliou, S. Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. *Comput. Netw.* **2021**, *195*, 108–177. [CrossRef]
5. Han, X.; Zhao, Y.; Yu, K.; Huang, X.; Xie, K.; Wei, H. Utility-Optimized Resource Allocation in Computing-Aware Networks. In Proceedings of the 2021 13th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 4–7 June 2021; pp. 199–205. [CrossRef]
6. Smeliansky, R. Network Powered by Computing. In Proceedings of the 2022 International Conference on Modern Network Technologies (MoNeTec), Moscow, Russia, 27–29 October 2022; pp. 1–5. [CrossRef]
7. Tang, X.; Cao, C.; Wang, Y.; Zhang, S.; Liu, Y.; Li, M.; He, T. Computing power network: The architecture of convergence of computing and networking towards 6G requirement. *China Commun.* **2021**, *18*, 175–185. [CrossRef]
8. Sun, Y.; Liu, J.; Huang, H.Y.; Zhang, X.; Lei, B.; Peng, J.; Wang, W. Computing Power Network: A Survey. *arXiv* **2022**, arXiv:2210.06080.
9. Kianpisheh, S.; Taleb, T. A Survey on In-Network Computing: Programmable Data Plane and Technology Specific Applications. *IEEE Commun. Surv. Tutorials* **2023**, *25*, 701–761. [CrossRef]
10. Zhang, J.; Xie, N.; Zhang, X.; Li, W. Multi-choice Virtual Machine Allocation with Time Windows in Cloud Computing. In *Green, Pervasive, and Cloud Computing: 13th International Conference, GPC 2018, Hangzhou, China, 11–13 May 2018*; Li, S., Ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 182–195.
11. Attaoui, W.; Sabir, E.; Elbiaze, H.; Sadik, M. Multi Objective Decision Making for Virtual Machine Placement in Cloud Computing. In *Network Games, Control and Optimization: 10th International Conference, NetGCooP 2020, Corsica, France, 22–24 September 2021*; Lasaulce, S., Mertikopoulos, P., Orda, A., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2021; pp. 154–166.
12. Fedorova, E.; Lapatin, I.; Lizyura, O.; Moiseev, A.; Nazarov, A.; Paul, S. Mathematical Modeling of Virtual Machine Life Cycle Using Branching Renewal Process. In Proceedings of the Information Technologies and Mathematical Modelling. Queueing Theory and Applications, Tomsk, Russia, 4–9 December 2023; Dudin, A., Nazarov, A., Moiseev, A., Eds.; Springer Nature Switzerland: Cham, Switzerland, 2023; pp. 29–39.
13. Cohen, J.; Kelly, F. A Paradox of Congestion in a Queuing Network. *J. Appl. Probab.* **1990**, *27*, 730–734. [CrossRef]
14. Knuth, D.E. *The Art of Computer Programming, Vols. 1–4*; Addison-Wesley: Reading, MA, USA, 2023.
15. Arora, S.; Barak, B. *Computational Complexity: A Modern Approach*; Cambridge University Press: Cambridge, UK, 2006.
16. Fortnow, L.; Homer, S. A Short History of Computational Complexity. *Bull. EATCS* **2003**, *80*, 95–133.
17. Dean, W. Computational Complexity Theory. In *The Stanford Encyclopedia of Philosophy*, Fall 2021 ed.; Zalta, E.N., Ed.; Metaphysics Research Lab., Stanford University: Stanford, CA, USA, 2021.
18. Sipser, M. *Introduction to the Theory of Computation*, 3rd ed.; Course Technology: Boston, MA, USA, 2013.
19. Borisov, A.; Gorshenin, A. Identification of Continuous-Discrete Hidden Markov Models with Multiplicative Observation Noise. *Mathematics* **2022**, *10*, 3062. [CrossRef]
20. Reuter, A. Methods for parallel execution of complex database queries. *Parallel Comput.* **1999**, *25*, 2177–2188. [CrossRef]
21. Ordonez, C.; Bellatreche, L. A Survey on Parallel Database Systems from a Storage Perspective: Rows Versus Columns. In *Database and Expert Systems Applications: 29th International Conference, DEXA 2018, Regensburg, Germany, 3–6 September 2018*; Elloumi, M., Granitzer, M., Hameurlain, A., Seifert, C., Stein, B., Tjoa, A.M., Wagner, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 5–20.
22. Kaeli, D.; Yew, P. *Speculative Execution in High Performance Computer Architectures*; Chapman & Hall/CRC Computer and Information Science Series; CRC Press: Boca Raton, FL, USA, 2005.
23. Liu, S.; Eisenbeis, C.; Gaudiot, J.L. Speculative Execution on GPU: An Exploratory Study. In Proceedings of the 2010 39th International Conference on Parallel Processing, San Diego, CA, USA, 13–16 September 2010; pp. 453–461. [CrossRef]
24. Estebanez, A.; Llanos, D.R.; Gonzalez-Escribano, A. A Survey on Thread-Level Speculation Techniques. *ACM Comput. Surv.* **2016**, *49*, 1–39. [CrossRef]
25. Goldreich, O. *Computational Complexity: A Conceptual Perspective*, 1st ed.; Cambridge University Press: New York, NY, USA, 2008.
26. Knuth, D.E. Big Omicron and Big Omega and Big Theta. *SIGACT News* **1976**, *8*, 18–24. [CrossRef]
27. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2001.
28. Kleinberg, J.; Tardos, E. *Algorithm Design*; Addison Wesley: Boston, MA, USA, 2006.
29. Bogdanov, A.; Trevisan, L. Average-Case Complexity. *Found. Trends Theor. Comput. Sci.* **2006**, *2*, 1–106. [CrossRef]
30. Ion, R.A.; Klaassen, C.A.J.; van den Heuvel, E.R. Sharp inequalities of Bienaymé–Chebyshev and Gauß type for possibly asymmetric intervals around the mean. *TEST* **2023**, *32*, 566–601. [CrossRef]
31. Borisov, A.; Bosov, A.; Ivanov, A. Application of Computer Simulation to the Anonymization of Personal Data: Synthesis-Based Anonymization Model and Algorithm. *Program. Comput. Softw.* **2023**, *49*, 730–734. [CrossRef]
32. Shah, B.; Bhavsar, H. Time Complexity in Deep Learning Models. *Procedia Comput. Sci.* **2022**, *215*, 202–210. [CrossRef]
33. Huber, P. *Robust Statistics*; Wiley: New York, NY, USA, 1981.
34. Nocedal, J.; Wright, S. *Numerical Optimization*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2006.

35. Gendreau, M.; Potvin, J.Y. (Eds.). *Handbook of Metaheuristics*, 2nd ed.; Springer: New York, NY, USA, 2010.
36. Koenker, R. *Quantile Regression*; Econometric Society Monographs; Cambridge University Press: Cambridge, UK, 2005.