UDC 004.738.5:004.722 **doi:** https://doi.org/10.20998/2522-9052.2024.3.09

Heorhii Kuchuk[1], Yevhen Kalinin[2], Nataliia Dotsenko[3], Igor Chumachenko[3], Yuriy Pakhomov[3]

[1] National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine
[2] National University of life and environmental sciences of Ukraine, Kyiv, Ukraine
[3] O.M. Beketov National University of Urban Economy in Kharkiv, Kharkiv, Ukraine

DECOMPOSITION OF INTEGRATED HIGH-DENSITY IoT DATA FLOW

Abstract. Topicality. The concept of fog computing made it possible to transfer part of the data processing and storage tasks from the cloud to fog nodes to reduce latency. But in batch processing of integrated data streams from IoT sensors, it is sometimes necessary to distribute the tasks of the batch between the fog and cloud layers. For this, it is necessary to decompose the formed package. But the existing methods of decomposition do not meet the requirements for efficiency in high-density IoT systems. **The subject of study** in the article are methods of decomposition of integrated data streams. **The purpose of the article** is to develop a method of decomposition of an integrated data stream in a dense high-density Internet of Things fog environment. This will reduce the processing time of operational transactions. **The following results** were obtained. The concept of decomposition of integrated information flows in the foggy layer was implemented to transition from the batch mode to the flow mode of task processing. Within the framework of the concept, a method of selecting elementary task flows from an integrated flow is proposed. An algorithm for decomposition of the integrated flow of tasks is proposed. **Conclusion.** A comparison of the proposed method of processing information flows in the foggy environment of high-density IoT with the existing approach is carried out. The results of the comparison showed that the proposed method is more suitable for deployment in conditions of limited network and computing resources. It is advisable to use it on nodes of fog computing systems with a high density of IoT sensors.

Keywords: Internet of Things; High Density IoT; Computer System; Fog Layer; Cloud Layer; Stream Processing; Batch Processing.

Introduction

The Internet of Things (IoT) demonstrates the rapid growth of technologies. The number of devices connected to the Internet is constantly growing [1]. The world market of the Internet of Things is constantly increasing [2]. The Internet of Things is used in many industries. There are examples of use in construction [3], industry [4, 5], state monitoring of complex systems [6], health care [7], etc.

The Internet of Things is based on cloud computing technology [8]. In recent years, difficulties have arisen as a result of the following factors [9]: the presence of the geographic distribution of IoT components; increase in network delays; high cost of communication channels; availability of mobile end devices.

Most of these difficulties were solved by introducing a layer of fuzzy computing [10]. Fog computing has brought data processing closer to the end devices of IoT networks. The concept of fog computing made it possible to transfer part of the data processing and storage tasks from the cloud to fog nodes to reduce latency [11]. But during batch processing of integrated data streams from IoT sensors, it is sometimes necessary to distribute the tasks of the batch between the fog and cloud layers [12].

When implementing batch processing, all input data must be pre-collected before starting the computing process. The most popular concept for forming an integrated data flow of High-density IoT is MapReduce. One of the most popular solutions to support the concept of MapReduce is Hadoop. Hadoop is a software platform for applications that provide reliable and fault-tolerant processing of big data, up to thousands of nodes. But the overheads associated with the decomposition of the integrated flow do not allow to ensure the operational processing of data in the foggy layer of High-density IoT.

Literature review. Let's consider some scientific works on this topic that can be applied to the decomposition of an integrated data stream.

In the article [13], decomposition is possible only for homogeneous flows. In articles [14, 15], the proposed decomposition method can be applied only in the cloud layer. The decomposition method proposed in article [16] is focused only on simple topologies. The decomposition algorithm proposed in the article [17] does not take into account the heterogeneity of the environment. Similar problems arise when applying the algorithm given in the article [18]. The algorithm given in the article [19] is not oriented to the real-time mode. The methods proposed in articles [20, 21] do not take into account the limited resources of the fog layer. Articles [22, 23] do not take into account the costs associated with data transmission.

Algorithms for decomposition of Internet of Things flows are proposed in articles [24, 25]. But it is used only in a homogeneous environment. The algorithms proposed in articles [26, 27] are focused only on structures similar to the structures of the cloud environment. The algorithm based on deep learning proposed in article [28] is effective only for a homogeneous environment, as is the algorithm used in article [29]. A decomposition strategy based on deep learning with the pooling of resource capabilities is proposed in the article [30]. But it cannot be implemented on a fog layer with limited device capabilities.

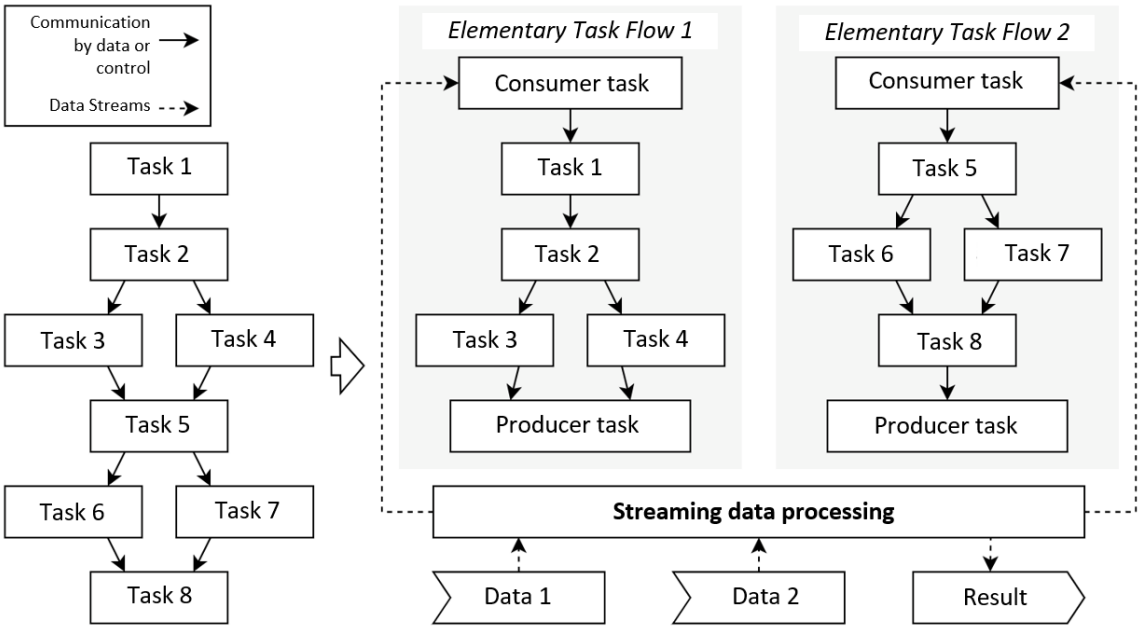
Therefore, the considered scientific works in the decomposition of the integrated data flow do not sufficiently take into account the characteristic features of the foggy environment of the high-density Internet of Things. Therefore, it is advisable to develop an appropriate method.

The purpose of the study. The purpose of the work is to develop a method of decomposition of an integrated data stream in a foggy environment of the high density

IoT. This will reduce the processing time of operational transactions.

1. Concept of decomposition of integrated information flow

Consider the Integrated Task Flow (ITF) of the High-density IoT fog environment. ITFs are formed for data processing in batch mode. ITF consists of many highly interconnected computational tasks [31]. We will decompose the ITF into Elementary Task Flow (ETF). Each ETF is a separate computing service. ETF can be transferred to another computing node. An individual ETF can be stopped independently of other ETFs. Communication between ETFs is provided using an event-oriented approach [32, 33]. This allows you to perform ITF in data streaming mode (Fig. 1).



a – ITF b – ETF

Fig. 1. Transition from batch job processing to stream processing

The proposed approach provides the following advantages:

– provides the ability to organize tasks in the data streaming mode;

– separates a strongly connected computing process in time and space; makes it possible to switch to the asynchronous communication model;

– provides an opportunity to independently develop, update, deploy and launch ETFs from the common task flow;

– provides the possibility of transparent integration of data flows coming from IoT devices;

– provides the possibility of deploying the flow of ETF tasks on nodes located at different levels of the fog computing system hierarchy; ETFs, which must provide low latency, can be deployed in fog nodes that are close to IoT devices; task streams that require long processing can be transferred to the cloud;

– provides the ability to move the ETF from one computing node to another without redistributing the computing load; at the same time, intermediate data is not lost and previous data does not need to be reprocessed.

The key steps in redistributing ITF to a set of ETFs are as follows:

– division of the integrated flow of tasks into sub-flows of tasks; each subthread contains part of the computational tasks of the basic task thread;

– formation of special peaks of consumers and message generators; message generators provide communication between elementary task flows and the data flow processing platform;

– creation of repositories in the data flow processing platform, which are responsible for organizing the reception and transmission of messages generated by elementary task flows;

– generation and orchestration of containers; containers provide encapsulation and the possibility of independent deployment of elementary task flows in the form of services.

2. Selection of Elementary Task Flows

ITF can be represented as a directed acyclic graph:

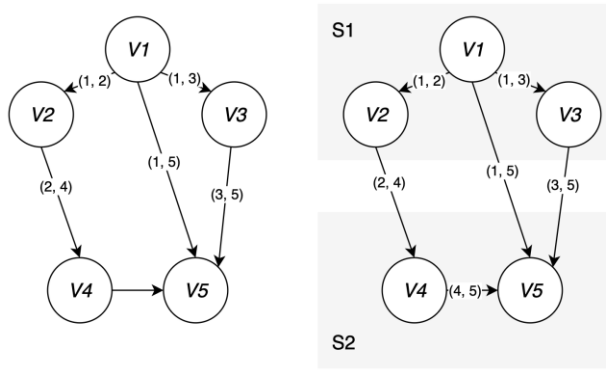
$$W = (V, E), \quad (1)$$

where W is integrated task flow; V is the set of vertices that constitute computational tasks (n is the total number of vertices); E is a set of edges connecting vertices that constitute data dependencies between tasks.

Each vertex $v_i \in V$ can have input and/or output edges.

The output degree $deg^+(v_i)$ of a vertex v_i in W is defined as the number of edges emanating from vi. The input degree deg^-v_i) of a vertex v_i in W is defined as the number of edges directed to v_i from other vertices. The edge $(v_i, v_j) \in E$ is a data dependence from v_i to v_j.

An example of decomposition of the integrated flow of tasks W into 2 subflows of tasks S_1 and S_2 is shown in Fig. 2.



a b

Fig. 2. An example of a task flow W (a, n = 5), b – dividing W into two task subflows S_1 and S_2

Consider the process of forming task subflows:

$$W = (S_1, \dots S_k), \quad S_i = (V_i, E_i), \tag{2}$$

where V_i is set of vertices included in a task subflow S_i, E_i is the set of edges between vertices included in V_i; the set of vertices V is divided by subflows of work into a set of disjoint subsets:

1) $\forall i \in \overline{1,k} \Rightarrow (V_i \in V; E_i \in E)$;

2) $\forall v \in V \Rightarrow \left(\exists i \in \overline{1,k} \middle| \, v \in V_i\right)$;

3) $E_i = \left\{(v_k, v_l) \in E; v_k, v_l \in V_i\right\}$;

4) $\forall i, j \Rightarrow \left(i \neq j \Rightarrow \left(V_i \cap V_j = \varnothing\right)\right)$.

Let's define the classes of edges and vertices associated with the task subflow S_i:

$$EI_i = \left\{(v_k, v_l) \in E \middle| \, v_k \notin S_i; v_l \in S_i\right\}; \tag{3}$$

$$EO_i = \left\{(v_k, v_l) \in E \middle| \, v_k \in S_i; v_l \notin S_i\right\}; \tag{4}$$

$$VI_i = \left\{\begin{matrix} v_l \in S_i \middle| (v_k, v_l) \in EI_i; \\ v_l \in \left\{v \middle| \deg^-(v) = 0\right\} \end{matrix}\right\}; \tag{5}$$

$$VO_i = \left\{\begin{matrix} v_k \in S_i \middle| (v_k, v_l) \in EO_i; \\ v_k \in \left\{v \middle| \deg^+(v) = 0\right\} \end{matrix}\right\}. \tag{6}$$

Let's mark:

cv_i *is consumer vertex;*

pv_i *is producer vertex*;

$ECV_i = \{(cv_i, v) : v \in VI_i\}$,

$EPV_i = \{(v, cp_i) : v \in VO_i\}$.

Then the elementary task flow ETF_i from the task subflow S_i is defined as

$$\mathrm{ET}F_i = (\mathrm{T}V_i, \mathrm{T}E_i), \tag{7}$$

where $\mathrm{T}V_i = V_i \cup \{cv_i, pv_i\}$; $\mathrm{T}E_i = E_i \cup ECV_i \cup EPV_i$.

The following data flow storages are required for the model to function:

DS_0 is responsible for the data that is needed to initialize the computational process in the task flow

– DS_i is responsible for data from ETFi , and their transfer to other tasks;

– DS_{out} is responsible for the resulting data.

In Fig. 3 shows the application of the elementary task flow model. The ETF1 block represents an elementary task flow. It includes the decision system vertex *cv*, the respondent vertex *pv*, computational tasks V_i.

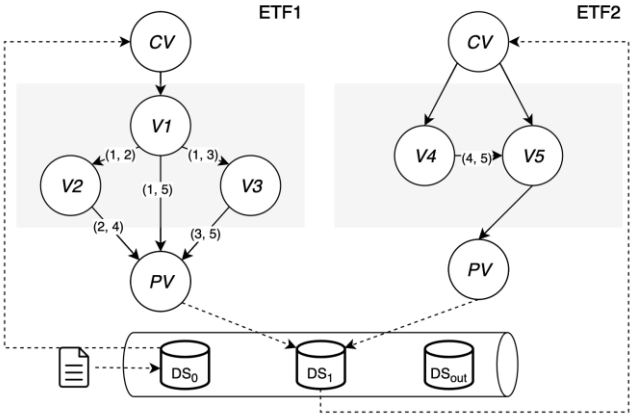


Fig. 3. The result of applying the elementary task flow model

The *ETF1* block, after completing its tasks, transfers the data to the DS1 storage for further processing. The *ETF2* block is an analogue of the *ETF1* block, but with its computational tasks V_i.

The arrows show the movement of data from the decision-making system to the execution of the direct computational process.

3. Algorithm of decomposition of IDF

Consider an algorithm that provides decomposition of an integrated data stream. The following key steps can be identified in the algorithm:

1. Initialization of the matrix $Z = Z_{n\times n}$ to describe the integrated data flow W with n vertices.

2. Mapping in matrix Z of internal and external edges for task subflows $Sx \in S$.

3. Formation of the matrix Mx to describe ETFx .

4. For each subflow of tasks $Sx \in S$, we will perform the following actions:

– initialization of matrix Mx;

– the mapping of the set of internal edges E_x into the matrix Mx;

– formation of the set of edges cv_x and their mapping into the matrix Mx;

– formation of the set of edges pv_x and their mapping into the matrix Mx.

We detail the steps of the algorithm. Let the ITF $W = (V, E)$ be divided into k subflows of tasks $S = (S1, \dots, Sk)$. Let's define the matrix $Z = Z_{n\times n}$:

$$Z = \begin{bmatrix} z_{1,1} & \cdots & z_{1,n} \\ \cdots & \cdots & \cdots \\ z_{n,1} & \cdots & z_{n,n} \end{bmatrix}. \tag{8}$$

The basis of the proposed approach is the modification of the graph representation mechanism using the adjacency matrix. The matrix Z will represent

the following information: presence of edges between vertices; to which subflow each vertex belongs.

The initialization of this matrix in the first step is implemented as follows:

$$z_{i,j} = \begin{cases} 1, & (i \neq j) \wedge \left((v_i, v_j) \in E \right); \\ 0, & (i \neq j) \wedge \left((v_i, v_j) \notin E \right); \\ x, & (i = j) \wedge (v_i \in V_x), \end{cases} \quad (9)$$

where $i, j \in 1..n$ is indices of matrix Z elements; $x \in 1..k$ is індекс формуємого підпотоку $Sx = (V_x, E_x)$.

In Fig. 4 shows an example of the initialization of the matrix Z for an example of a task flow W from two subflows.

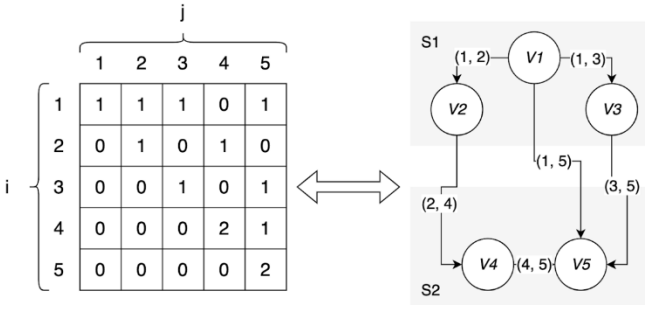


Fig. 4. An example of matrix initialization Z

In the second step of the algorithm, internal and external edges are determined for each subflow in Z:

$$z_{i,j} = \begin{cases} z_{i,i}, & (i \neq j) \wedge \left(z_{i,j} = 1 \right) \wedge \left(z_{i,i} = z_{j,j} \right); \\ -1, & (i \neq j) \wedge \left(z_{i,j} = 1 \right) \wedge \left(z_{i,i} \neq z_{j,j} \right); \\ z_{i,j}, & else. \end{cases} \quad (10)$$

Consider the case when the values of $z_{i,i}$ and $z_{j,j}$ for the edge represented by $z_{i,j}$ are the same. This means that vertices v_i and v_j are in the same subflow. Then the value of $z_{i,j}$ is set equal to the index value of the corresponding subflow. This value is stored in the corresponding diagonal elements of the matrix Z.

In the second case, the values of the corresponding diagonal elements $z_{i,i}$ and $z_{j,j}$ differ. They characterize the vertices v_i and v_j differently, that is, the vertices are in different subflows. Therefore, (v_i, v_j) is an external edge. In this case, the value of the edge $z_{i,j}$ is set equal to "–1". In Fig. 5 shows an example of applying step 2. The result of this step is the formed matrix Z.

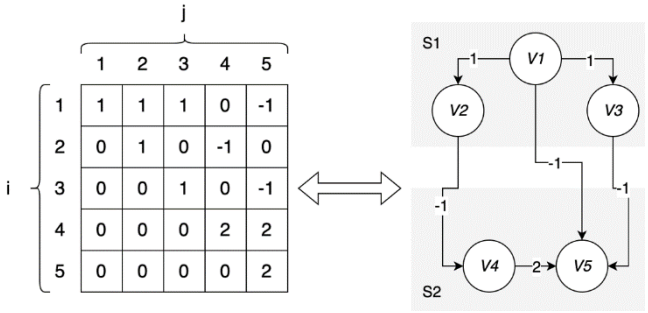


Fig. 5. Inner and outer edges of subflows

In the next step of the algorithm, the matrices Mx are initialized. They represent elementary ETFx task threads based on Sx subthreads. We define Mx as the matrix used to determine ETFx based on Sx and vertices cvx and pvx. To index the rows and columns of the elements in the Mx matrices, we will use the variables a and b, respectively (Fig. 6).

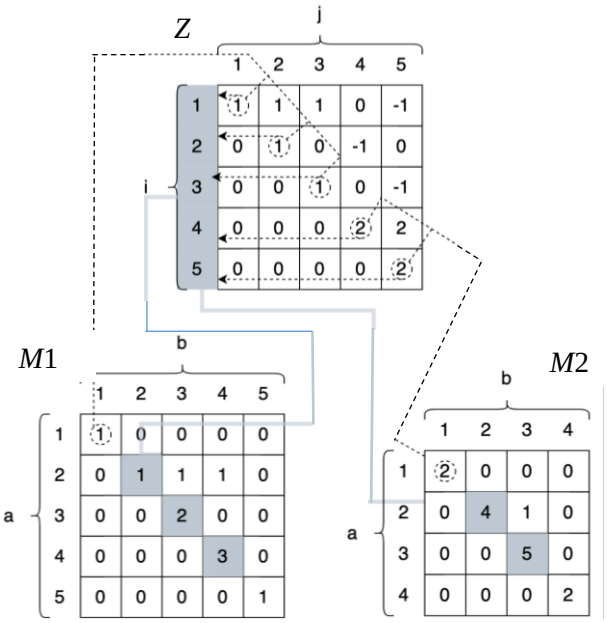


Fig. 6. Initialization of matrices M_1 and M_2

The dimensionality r_x of the matrix Mx is determined by the number of rows in the matrix Z, in which the values on the diagonal are equal to x:

$$r_x = \dim \left\{ a \in \overline{1,n} \,\middle|\, z_{a,a} = x \right\}. \quad (11)$$

To represent the vertexs cv_x and pv_x, it is necessary to add 2 rows and 2 columns to the matrix Mx as follows:

$$Mx = \begin{bmatrix} mx_{1,1} & \cdots & mx_{1,r_x+2} \\ \cdots & \cdots & \cdots \\ mx_{m_{x_{1,1}},1} & \cdots & mx_{r_x+2,r_x+2} \end{bmatrix}. \quad (12)$$

The node representation cv_x will be placed at position $mx_{1,1}$ and pv_x at position $mx_{r+2,\,r+2}$. The diagonal values in the Mx of both these nodes are set equal to x. The rest of the diagonal values store the indices i of the task flow vertices W, which have a value equal to x. Let's save the values of indices i of all vertices from Sx in D_x:

$$D_x = \left\{ z_{i,i} = x \quad \forall i \in \overline{1,n} \right\}. \quad (13)$$

The values of $d \in D_x$ will be used to fill the diagonal values for the matrix Mx:

$$mx_{a.a} = \begin{cases} d_{a-1}, & 1 < a < r_x + 2; \\ x, & (a = 1) \vee (a = r_x + 2), \end{cases} \quad (14)$$

where $x \in 1..k$ is subflow index Sx; $a \in 1..r_x + 2$ is the row index in the matrix Mx.

After the initialization of the matrices Mx for each subflow Sx, the representation of all internal edges of the

subflows Ex is implemented in them. In order to transfer the internal edges of Ex from Z to Mx, the following formula is used:

$$mx_{a,b} = \begin{cases} 1, & z_{mx_{a,a}\,mx_{b,b}}; \\ 0, & else; \end{cases} \qquad (15)$$

$$\forall mx_{a,b} \in Mx : \left(a,b \in \overline{2, r_x + 1}\right) \wedge \left(a \neq b\right).$$

In this case, the adjacent vertex information stored in the diagonal positions of Mx is used as a reference, where a is the row index in Mx and b is the column index in Mx. In Fig. 7 shows an example transfer the inner edge (v_1, v_2) from $E1$ to $M1$. We also see the results of the representation of internal edges of task subflows from sets $E1$ and $E2$ in $M1$ and $M2$, respectively.

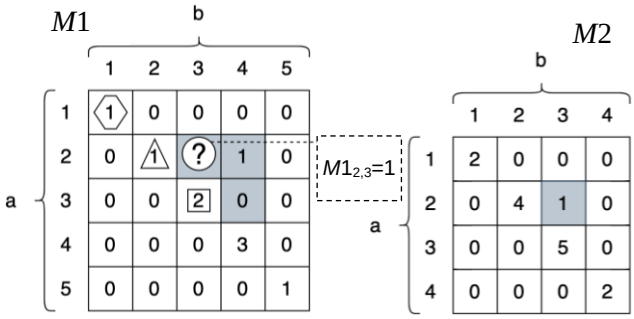


Fig. 7. The results of applying the formula (15)

After presenting all internal edges from Ex to Mx, output edges are determined. There are 2 types of vertices in Mx that must receive input edges from cv_x: is the initial vertex W, which has no incoming edges; vertices that receive input edges from vertices in other subflows.

Let's calculate the values of the positions in the first row Mx for the outgoing edges from the vertex cv_x:

$$mx_{1,b} = \left(\forall mx_{1,b} \in Mx : b \in \overline{2, r_x + 1}\right) =$$

$$= \begin{cases} 1, & \left(\left(\forall i \in \overline{1,n}\right) \wedge \left(i \neq mx_{b,b}\right) \middle| z_{i,mx_{b,b}} = 0\right) \wedge \\ & \wedge \left(\left(\exists i \in 1..n\right) : z_{i,mx_{b,b}} = -1\right); \\ 0, & else. \end{cases} \qquad (16)$$

In Fig. 8 shows an example of determining the presence of an edge that should be represented in position $m1_{1.2}$ in $M1$.

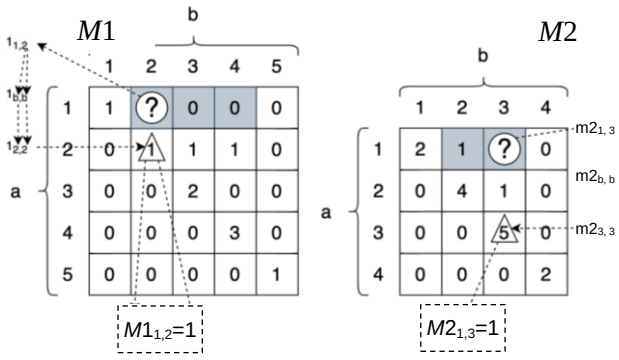


Fig. 8. An example of a connection cv

The second case for the value "1" defines edges from cv_x to those vertices that have input edges from other subflows. Similarly, the value $mx_{b,b}$ is used to obtain the index of the column searched in the Z matrix.

The presence of at least one element with the value "1" in the corresponding column means that this vertex receives data from an external subthread. This vertex must receive an input edge from cv_x to Mx. In Fig. 8 shows an example of the application of this case to determine the presence of an edge, which should be represented in the position $m2_{1.3}$ in $M2$.

The final step of the redistribution algorithm is to determine the edges to be completed at the generator vertex pvx for each subflow matrix Mx.

There are 2 types of vertices in Sx, which must be connected by edges to pv_x: a vertex that has no outgoing edges in Ex; vertices whose output edges go to vertices located in other elementary subflows.

The values of the positions in the column (r_x + 2) of the Mx are as follows:

$$mx_{a,r_x+2} = \left(\forall mx_{a,r_x+2} \in Mx : a \in \overline{2, r_x + 1}\right) =$$

$$= \begin{cases} 1, & \left(\left(\forall b \in \overline{2, r_x + 1}\right) \wedge \left(a \neq b\right) : mx_{a,b} = 0\right) \wedge \\ & \wedge \left(\left(\exists j \in \overline{1,n}\right) : z_{mx_{a,j}} = -1\right); \\ 0, & else. \end{cases} \qquad (17)$$

The first case for the value "1" defines those vertices that do not have outgoing edges from the very beginning. Such vertices are connected by edges with pv_x. In Fig. 9 shows an example of the application of this case to determine the presence of an edge, which should be represented in the position $m2_{2.4}$ in $M2$.

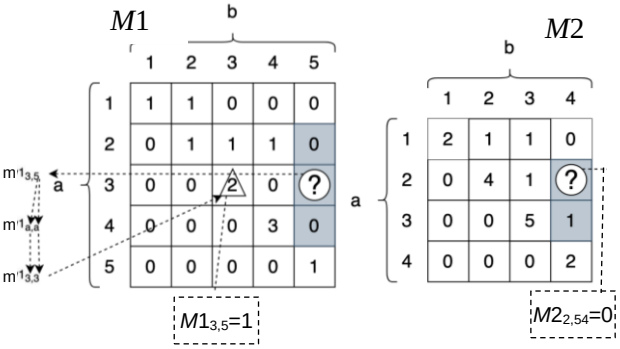


Fig. 9. An example of a connection pv

The second case for the value "1" defines the edges in pvx from those vertices that originally had outgoing edges in the vertices of other subflows. The value $mx_{a,a}$ is used to obtain the index of the row by which elements in the matrix are searched.

The presence of at least one element with the value "1" in the row with the index $mx_{a,a}$ in the matrix Z means that this vertex transmits data to the external subflow. In this case, this vertex must be connected by an outgoing edge from pvx to Mx.

In Fig. 9 shows an example of the application of this case to determine the presence of an edge, which should be represented in the position $m1_{3.5}$ in $M1$.

Therefore, the integrated task flow *W* is divided into elementary task flows *MIF* 1 and *MIF* 2. In Fig. 10 shows the final states of matrices *M*1 and *M*2 representing these elementary task flows.

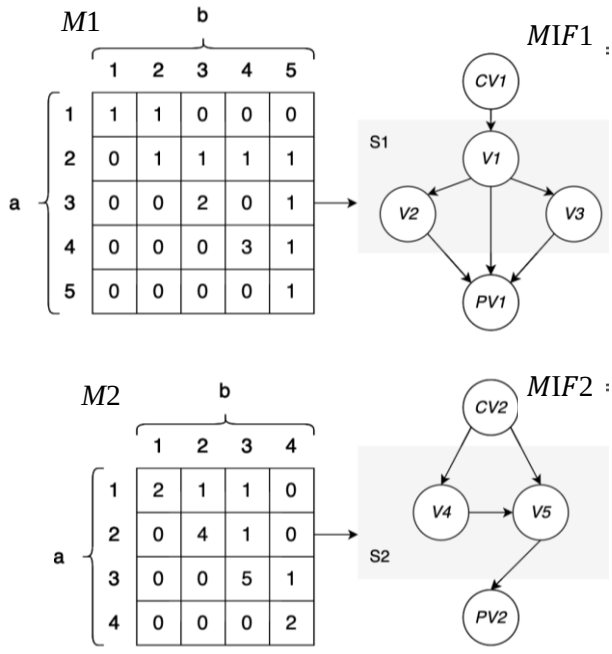


Fig. 10. Final matrices *M*1 and *M*2
of elementary task flows *MIF* 1 and *MIF* 2

4. Discussion of results

A comparison of the proposed method of processing information flows in a foggy high-density IoT environment with the existing approach was carried out. The criterion is the average reaction time to an event, information about which can be obtained from the data stream in real time. For comparison, a high-density IoT sensor simulator was generated. The results of the comparison are given in Table 1.

Table 1 – **Comparison of methods by response time to critical messages**

Time, min	***N* × *10*$^{-2}$**	***N*0**	***t*, microsec**	
			A	***Y***
IoT				
1	1	2	4,3	6,5
5	6	7	7,9	9,7
10	11	14	9,6	11,8
30	36	51	10,6	13,9
60	58	98	11,2	14,9
High-density IoT				
1	50	28	9,6	14,3
5	289	121	51,2	21,3
10	584	237	112,7	25,5
30	1766	722	287,5	32,2
60	2984	1325	522,9	48,7

In the first line of the table, the simulation time is indicated.

In the Table 1, the following designations are accepted:

N is the number of messages received by the fog layer from IoT sensors;

*N*0 – the number of events associated with a change in the state of IoT sensors;

t is the average reaction time of the system to the event of a sensor state change;

A is standard method;

Y is the proposed method.

The results obtained in this experiment allow us to draw the following conclusions.

When processing High-density IoT data using elementary task flows, the average response time was significantly lower than the response time provided by the integrated task flow. This is achieved by streaming results as soon as they become available. The integrated thread waits for the data preparation phase to complete before processing it. But for ordinary IoT, streaming processing does not have a significant advantage. In addition, the time to obtain results increases because decomposition must be performed.

The average event response time when processing in streaming mode does not depend on the total execution period of the batch.

When implementing data processing in batch mode, the average response time increases as the size of the input data increases.

Conclusions

The article considers an approach to reducing the response time for high-density IoT operational tasks. To do this, it is proposed to process tasks in the flow mode of task processing. The concept of decomposition of integrated information flows in the foggy layer has been implemented to move from the batch mode to the flow mode of task processing. As part of the concept, a method of selecting elementary task flows from an integrated flow is proposed. An algorithm for decomposition of the integrated flow of tasks is proposed.

A comparison of the proposed method of processing information flows in the foggy environment of high-density IoT with the existing approach is carried out. The results of the comparison showed that the proposed method is more suitable for deployment in conditions of limited network and computing resources. It is advisable to use it on nodes of fog computing systems with a high density of IoT sensors. Then you can achieve a reaction to operational events in a mode close to real time:

Acknowledgements

The study was funded by the National Research Foundation of Ukraine in the framework of the research project 2022.01/0017 on the topic “Development of methodological and instrumental support for Agile transformation of the reconstruction processes of medical institutions of Ukraine to overcome public health disorders in the war and post-war periods”.

REFERENCES

1. Schulz, A.S. (2023), “User Interactions with Internet of Things (IoT) Devices in Shared Domestic Spaces”, *ACM International Conference Proceeding Series*, pp. 577–579. doi: https://doi.org/10.1145/3626705.3632615

2. Pardo, C., Wei, R. and Ivens, B.S. (2022), "Integrating the business networks and internet of things perspectives: A system of systems (SoS) approach for industrial markets", *Industrial Marketing Management*, vol. 104, pp. 258–275, doi: https://doi.org/10.1016/j.indmarman.2022.04.012
3. Zhang, Z. (2023), "A computing allocation strategy for Internet of things' resources based on edge computing", *International Journal of Distributed Sensor Networks*, vol. 17(12), doi: https://doi.org/10.1177/15501477211064800
4. Chalapathi, G.S.S., Chamola, V., Vaish, A. and Buyya, R. (2022), "Industrial internet of things (Iiot) applications of edge and fog computing: A review and future directions", *Advances in Information Security*, vol. 83, pp. 293–325, doi: https://doi.org/10.1007/978-3-030-57328-7_12
5. Dotsenko, N., Chumachenko, I., Galkin, A., Kuchuk, H. and Chumachenko, D. (2023), "Modeling the Transformation of Configuration Management Processes in a Multi-Project Environment", *Sustainability (Switzerland)*, Vol. 15(19), 14308, doi: https://doi.org/10.3390/su151914308
6. Zuev, A., Karaman, D. and Olshevskiy, A. (2023), "Wireless sensor synchronization method for monitoring short-term events", Advanced Information Systems, vol. 7, no. 4, pp. 33–40, doi: https://doi.org/10.20998/2522-9052.2023.4.04
7. Krishnan, S. and Ilmudeen, A. (2023), "Internet of Medical Things in Smart Healthcare: Post-COVID-19 Pandemic Scenario", *Imprint Apple Academic Press*, New York, doi: http://dx.doi.org/10.1201/9781003369035
8. Fatlawi, A., Al Dujaili, M.J. (2023), Integrating the Internet of Things (IoT) and Cloud Computing Challenges and Solutions: A Review. AIP Conference Proceedings, 2977(1), 020067. doi: http://dx.doi.org/10.1063/5.0181842
9. Qayyum, T., Trabelsi, Z., Waqar Malik, A. and Hayawi, K. (2022), "Mobility-aware hierarchical fog computing framework for Industrial Internet of Things", *Journal of Cloud Computing*, vol. 11(1), doi: https://doi.org/10.1186/s13677-022-00345-y
10. Kuchuk, H. and Malokhvii, E. (2024), "Integration of IOT with Cloud, Fog, and Edge Computing: A Review", *Advanced Information Systems*, vol. *8*(2), pp. 65–78, doi: https://doi.org/10.20998/2522-9052.2024.2.08
11. Kuchuk, N., Mozhaiev, O., Semenov, S., Haichenko, A., Kuchuk, H., Tiulieniev, S., Mozhaiev, M., Davydov, V., Brusakova, O. and Gnusov, Y. (2023). Devising a method for balancing the load on a territorially distributed foggy environment. Eastern-European Journal of Enterprise Technologies, 1(4 (121), 48–55. https://doi.org/10.15587/1729-4061.2023.274177
12. Hunko, M., Tkachov, V., Kuchuk, H. and Kovalenko, A. (2023), Advantages of Fog Computing: A Comparative Analysis with Cloud Computing for Enhanced Edge Computing Capabilities, *2023 IEEE 4th KhPI Week on Advanced Technology, KhPI Week 2023 – Conf. Proc*, 02-06 October 2023, Code 194480, doi: https://doi.org/10.1109/KhPIWeek61412.2023.10312948
13. Lu, S., Wu, J., Wang, N., Duan, Y., Liu, H., Zhang, J. and Fang, J. (2023), "Resource provisioning in collaborative fog computing for multiple delay-sensitive users", *Software – Practice and Experience*, vol. 53, is. 2, pp. 243–262, doi: https://doi.org/10.1002/spe.3000
14. Petrovska, I. and Kuchuk, H. (2023), "Adaptive resource allocation method for data processing and security in cloud environment", *Advanced Information Systems*, vol. 7, no. 3, pp. 67–73, doi: https://doi.org/10.20998/2522-9052.2023.3.10
15. Kuchuk, G., Nechausov, S. and Kharchenko, V. (2015), "Two-stage optimization of resource allocation for hybrid cloud data store", *Int. Conf. on Information and Digital Techn*, Zilina, pp. 266–271, doi: http://dx.doi.org/10.1109/DT.2015.7222982
16. Li, G., Liu, Y., Wu, J., Lin, D. and Zhao, Sh. (2019), "Methods of Resource Scheduling Based on Optimized Fuzzy Clustering in Fog Computing", *Sensors*, MDPI, vol. 19(9), doi: https://doi.org/10.3390/s19092122
17. Jamil, B. Shojafar, M., Ahmed, I., Ullah, A., Munir, K. and Ijaz, H. (2020), "A job scheduling algorithm for delay and performance optimization in fog computing", *Concurrency and Computation: Practice and Experience*, vol. 32(7), doi: https://doi.org/10.1002/cpe.5581
18. Gomathi, B., Saravana Balaji, B., Krishna Kumar, V., Abouhawwash, M., Aljahdali, S., Masud, M. and Kuchuk, N. (2022), "Multi-Objective Optimization of Energy Aware Virtual Machine Placement in Cloud Data Center", Intelligent Automation and Soft Computing, Vol. 33(3), pp. 1771–1785, doi: http://dx.doi.org/10.32604/iasc.2022.024052
19. Proietti Mattia, G. and Beraldi, R. (2023), "P2PFaaS: A framework for FaaS peer-to-peer scheduling and load balancing in Fog and Edge computing", *SoftwareX*, vol. 21, doi: https://doi.org/10.1016/j.softx.2022.101290
20. Kovalenko, A. and Kuchuk, H. (2022), "Methods to Manage Data in Self-healing Systems", *Studies in Systems, Decision and Control*, vol. 425, pp. 113–171, doi: https://doi.org/10.1007/978-3-030-96546-4_3
21. Kuchuk, N., Kovalenko, A., Ruban, I., Shyshatskyi, A., Zakovorotnyi, O. and Sheviakov, I. (2023), "Traffic Modeling for the Industrial Internet of NanoThings", *2023 IEEE 4th KhPI Week on Advanced Technology*, KhPI Week 2023 - Conference Proceedings, 2023, doi: 194480. http://dx.doi.org/10.1109/KhPIWeek61412.2023.10312856
22. Sharma, Sh. Saini H. (2019), "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment", *Sustainable Computing: Informatics and Systems*, vol. 24, doi: https://doi.org/10.1016/j.suscom.2019.100355
23. Attar, H., Khosravi, M.R., Igorovich, S.S., Georgievan, K.N. and Alhihi, M. (2020), "Review and performance evaluation of FIFO, PQ, CQ, FQ, and WFQ algorithms in multimedia wireless sensor networks", *International Journal of Distributed Sensor Networks*, vol. 16(6), doi: https://doi.org/10.1177/1550147720913233
24. Malik, U.M., Javed, M.A., Frnda, J., Rozhon, J. and Khan, W.U. (2022), "Efficient Matching-Based Parallel Task Offloading in IoT Networks", *Sensors*, vol. 22, doi: https://doi.org/10.3390/s22186906
25. Liu, L., Chen, H., Xu, Z. (2022). SPMOO: A Multi-Objective Offloading Algorithm for Dependent Tasks in IoT Cloud-Edge-End Collaboration. Information, 13, 75. doi: https://doi.org/10.3390/info13020075
26. Ghenai, A., Kabouche, Y. and Dahmani, W. (2018), "Multi-user dynamic scheduling-based resource management for Internet of Things applications", *2018 International Conference on Internet of Things, Embedded Systems and Communications* (IINTEC), doi: https://doi.org/10.1109/IINTEC.2018.8695308
27. Kuchuk, G.A., Akimova, Yu.A. and Klimenko, L.A. (2000), "Method of optimal allocation of relational tables", *Engineering Simulation*, 2000, vol. 17(5), pp. 681–689, available at: https://www.scopus.com/record/display.uri?eid=2-s2.0-0034512103&origin=resultslist&sort=plf-f#metrics
28. Wei, J.-Y. and Wu, J.-J. (2023), "Resource Allocation Algorithm in Industrial Internet of Things Based on Edge Computing", *Journal of Northeastern University*, vol. 44(8). doi: https://doi.org/10.12068/j.issn.1005-3026.2023.08.002
29. Yaloveha, V., Podorozhniak, A. and Kuchuk, H. (2022), "Convolutional neural network hyperparameter optimization applied to land cover classification", *Radioelectronic and Computer Systems*, vol. 1(2022), pp. 115–128, doi: https://doi.org/10.32620/reks.2022.1.09

30. Zhang, Z. (2023), “A computing allocation strategy for Internet of things’ resources based on edge computing”, *International Journal of Distributed Sensor Networks*, vol. 17(12), doi: https://doi.org/10.1177/15501477211064800
31. Petrovska, I., Kuchuk, H., Kuchuk, N., Mozhaiev, O., Pochebut, M. and Onishchenko, Yu. (2023), “Sequential Series-Based Prediction Model in Adaptive Cloud Resource Allocation for Data Processing and Security”, *2023 13th International Conference on Dependable Systems, Services and Technologies, DESSERT 2023*, 13–15 October, Athens, Greece, code 197136, doi: https://doi.org/10.1109/DESSERT61349.2023.10416496
32. Kalinin, Y., Kozhushko, A., Rebrov, O. and Zakovorotniy, A. (2022), “Characteristics of Rational Classifications in Game-Theoretic Algorithms of Pattern Recognition for Unmanned Vehicles”, *2022 IEEE 3rd KhPI Week on Advanced Technology* (KhPIWeek), Kharkiv, Ukraine, pp. 1-5, doi: https://doi.org/10.1109/KhPIWeek57572.2022.9916454
33. Pisching, M., Pessoa, M.A.O., Junqueira, F., Filho, D. J. S. and Miyagi, P. E. (2018), “An architecture based on RAMI 4.0 to discover equipment to process 168 operations required by products”, *Computers & Industrial Engineering*, vol. 125, pp. 574–591, doi: https://doi.org/10.1016/j.cie.2017.12.029

Received (Надійшла) 19.05.2024
Accepted for publication (Прийнята до друку) 14.08.2024

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Кучук Георгій Анатолійович – доктор технічних наук, професор, професор кафедри комп’ютерної інженерії та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Heorhii Kuchuk – Doctor of Technical Sciences, Professor, Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: kuchuk56@ukr.net; ORCID Author ID: http://orcid.org/0000-0002-2862-438X;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57057781300.

Калінін Євген Іванович – доктор технічних наук, професор, завідувач кафедри тракторів і автомобілів, Національний університет біоресурсів і природокористування України, Київ, Україна;
Yevhen Kalinin – Doctor of Technical Sciences, Professor, Professor of Tractors and automobiles Department, National University of life and environmental sciences of Ukraine, Kyiv, Ukraine;
e-mail: kalininhntusg@gmail.com; ORCID Author ID: http://orcid.org/0000-0001-6191-8446;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57204677409.

Доценко Наталія Володимирівна – доктор технічних наук, професор, професорка кафедри управління проєктами в міському господарстві та будівництві, Харківський національний університет міського господарства, Харків, Україна;
Nataliia Dotsenko – Doctor of Technical Sciences, professor, Professor of Project Management in Urban Management and Construction Department, O.M. Beketov National University of Urban Economy in Kharkiv, Kharkiv, Ukraine;
e-mail: nvdotsenko@gmail.com; ORCID Author ID: http://orcid.org/0000-0003-3570-5900;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57204939770.

Чумаченко Ігор Володимирович – доктор технічних наук, професор, завідувач кафедри управління проєктами в міському господарстві та будівництві, Харківський національний університет міського господарства, Харків, Україна;
Igor Chumachenko – Doctor of Technical Sciences, professor, Head of Project Management in Urban Management and Construction Department, O.M. Beketov National University of Urban Economy in Kharkiv, Kharkiv, Ukraine;
e-mail: ivchumachenko@gmail.com; ORCID Author ID: http://orcid.org/0000-0003-2312-2011;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57194419994.

Пахомов Юрій Васильович – гандидат технічних наук, доцент, доцент кафедри комп’ютерних наук та інформаційних технологій, Харківський національний університет міського господарства, Харків, Україна;
Yuriy Pakhomov – Candidate of Technical Sciences, Associate Professor, Associate Professor of Department of Computer Science and Information Technologies, O.M. Beketov National University of Urban Economy in Kharkiv, Kharkiv, Ukraine;
e-mail: abc050073@gmail.com; ORCID Author ID: http://orcid.org/0000-0002-2267-8600;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57190816915.

Декомпозиція інтегрованого потоку даних IoT великої щільності

Г. А. Кучук, Є. І. Калінін, Н. В. Доценко, І. В. Чумаченко, Ю. В. Пахомов

Анотація. Актуальність. Концепція туманних обчислень дозволила перенести частину завдань по обробці і зберіганню даних з хмари на туманні вузли для зниження затримки. Але при пакетній обробці інтегрованих потоків даних від сенсорів IoT іноді необхідно проводити розподіл завдань пакету між туманним і хмарним шарами. Для цього необхідно провести декомпозицію сформованого пакету. Але існуючі методи декомпозиції не відповідають вимогам по оперативності у системах Інтернету речей великої щільності. **Предметом вивчення** в статті є методи декомпозиції інтегрованих потоків даних. **Метою статті** є розробка методу декомпозиції інтегрованого потоку даних у туманному середовищі Інтернету речей великої щільності. Це дозволить зменшити час обробки оперативних транзакцій. Отримано **наступні результати**. Для переходу від пакетного режиму потокового режиму обробки завдань реалізовано концепцію декомпозиції інтегрованих інформаційних потоків у туманному шарі. В рамках концепції запропоновано метод виділення елементарних потоків завдань із інтегрованого потоку. Запропоновано алгоритм декомпозиції інтегрованого потоку завдань. **Висновок.** Проведено порівняння запропонованого методу обробки інформаційних потоків у туманному середовищі IoT великої щільності із існуючим підходом. Результати порівняння показали, що запропонований метод більше підходить для розгортання в умовах обмежених мережних та обчислювальних ресурсів. Його доцільно використовувати на вузлах туманних обчислювальних систем при великій щільності датчиків IoT.

Ключові слова: Інтернет речей; IoT великої щільності; комп’ютерна система; туманний шар; хмарний шар; потокова обробка; пакетна обробка.