DOI: 10.17725/rensit.2023.15.179

The problems of scaling components in streaming data processing systems

George G. Bulychev, Alexey V. Chernykh
MIREA-Russian Technological University, https://www.mirea.ru/
Moscow 119454, Russian Federation
E-mail: geo-bulychev@mail.ru, meidm@yandex.ru

Received April 24, 2023, peer-reviewed April 30, 2023, accepted May 08, 2023

Abstract: **The article examines the existing problems of scaling components in streaming data processing systems. The algorithm proposed by the authors significantly reduces the number of operations for creating and removing components when scaling the system. The algorithm is based on linear regression. For the proposed algorithm, the practical load on the system is simulated to confirm the theoretical results obtained.**

Keywords: **streaming data processing, scaling, big data**

UDC 004.62

For citation: George G. Bulychev, Alexey V. Chernykh. The problems of scaling components in streaming data processing systems. *RENSIT: Radioelectronics. Nanosystems. Information Technologies*, 2023, 15(2):179-184e. DOI: 10.17725/rensit.2023.15.179.

Contents

1. INTRODUCTION

In the conditions of the modern information society, characterized by gigantic volumes of data [1], constantly arriving in real time, the relevance and importance of streaming data processing [2] is steadily growing. This approach to information processing involves immediate analysis and processing of data, unlike the classical approach with data storage and post-processing.

The main advantage of streaming data processing systems is the ability to quickly make decisions based on fresh and up-to-date information, which allows organizations and specialists to adapt to dynamically changing conditions and maintain competitiveness in the market. However, along with the growing scale and complexity of data processing flows, a number of technical and conceptual problems arise. In particular, algorithms for scaling components in streaming data processing systems face requirements related to efficiency, reliability and flexibility.

The need to quickly and efficiently allocate resources between data processing nodes creates problems in determining the optimal amount of resources for each node to ensure high performance and reliability of the system. Incorrect scaling can lead to reduced performance, additional resource costs, and difficulty processing data in real time.

2. ANALYSIS OF EXISTING ALGORITHMS

Existing streaming data processing systems use a scaling algorithm with a threshold value [3] due to the high decision-making speed.

2.1. Scaling algorithm with a threshold value

The scaling algorithm with a threshold value is based on determining the threshold values of the load on the system nodes. If the load on a node exceeds a certain threshold, the system automatically scales to distribute the load.

The result of the algorithm is the number of component entities - a value that characterizes the required number of copies of the component to process the incoming load.

The number of component entities is calculated using the formula:

$$C_t = N_{t-1}/M, \quad (1)$$

where C_t – is the number of component entities at time t, N_{t-1} – is the load on the system at time t - 1, M – is the threshold value of the load.

As can be seen from the formula, the algorithm does not predict the load and operates only with known values. This feature leads to a "lag" of the algorithm from the current load. With a jumpy load schedule, the system will adjust to the load with a lag, performing an excessive number of operations for creating and deleting component entities.

3. LOAD FORECASTING ALGORITHMS

To scale components efficiently, it is necessary to reduce the number of operations for creating and deleting component entities. To fulfill this requirement, it is necessary to implement a load prediction algorithm. The problem of load forecasting can be reduced to the problem of time series forecasting (since the load is directly related to the time series).

However, in the context of an existing task, the algorithm should have a minimum decision delay time and use a minimum number of resources. Otherwise, the algorithm will work with a strong delay or with a significant increase in resource consumption, thereby leveling the savings caused by a decrease in the number of operations for creating or deleting component entities.

These features do not allow using algorithms based on: neural networks [4], random forest [5] and most other machine learning algorithms [6,7]. However, the linear regression algorithm [8] and exponential smoothing [9] fall under these requirements.

3.1. Linear regression algorithm

Linear regression is a statistical machine learning method used to model the relationship between a dependent variable and one or more independent variables. In the context of load forecasting, the dependent variable can be the load on the system, and the independent variables are factors that affect the load (for example, time of day, day of the week, traffic volume).

Mathematically, linear regression is described as:

$$C_{t+1} = \frac{\overline{Y}_{t+1}}{M}, \quad \overline{Y}_{t+1} = \delta \cdot N_t + \varepsilon, \quad (2)$$

where C_{t+1} – is the number of component entities at time $t + 1$, $\overline{Y}_{t+1}$ – is the predicted

load at time $t + 1$, M – is the threshold value of the load, N_t – load on the system at time t, δ,ε – regression coefficients.

The linear regression coefficients are selected using the least squares method.

3.2. Exponential smoothing algorithm

Exponential smoothing is a time series forecasting method that takes into account all observations in the past by assigning exponentially decreasing weights to them. Thus, newer observations have a greater impact on the forecast than older ones.

To solve the existing problem, the algorithm of triple exponential smoothing (the Holt-Winters method [10]) will be optimal, since this algorithm takes into account trends and seasonality, and most of the processed messages are characterized by these features.

The mathematical algorithm is described as:

$$\begin{aligned} &C_{t+h} = \frac{\overline{Y}_{t+h}}{M}, \\ &\overline{Y}_{t+h} = A(t) + h \cdot B(t) + S(t - p + 1 + (h-1) \bmod p), \\ &A(t) = \alpha \cdot (N_t - S(t-p)) + (1-\alpha) \cdot (A(t-1) + B(t-1)t), \\ &B(t) = \beta \cdot (A(t) - A(t-1)) + (1-\beta) \cdot B(t-1), \\ &S(t) = \gamma \cdot (N_t - A(t)) + (1-\gamma) \cdot S(t-p), \end{aligned} \quad (3)$$

where C_{t+h} – is the number of component entities at time $t + h$, $\overline{Y}_{t+h}$ – is the predicted load at time $t + 1$, M – is the threshold value of the load, N_t – is the load on the system at time t, A – is the equation describing the smoothed series, B – is the equation for estimating the trend, S – is the equation for estimating seasonality, α – is the constant value determining the effect of the smoothed series, β – is the constant value determining the effect of the trend, γ – is the constant values determining the effects of seasonality, p – the period of seasonality.

For the algorithm to function, it is necessary to determine constant values. However, for most components it is impossible to estimate them in advance.

4. MODELING

4.1. Key indicators

Using the algorithm should lead to:

- Reducing the number of operations for creating and deleting components.
- Reducing the delay in processing messages from the moment they arrive in the line. For streaming data processing systems, the critical parameter is the processing time of each message.
- Reducing the maximum and average line size. The system is modeled using the example of one component, however, in the target system, the number of components and lines can be in the thousands, and with a significant increase in the line size of each component, the total memory consumption of the system can offset the resource savings obtained by reducing the number of scaling operations.

The target algorithm should show the minimum value of the specified parameters during modeling.

The algorithm should use minimal resources to predict the load.

To evaluate all of the above parameters, the following metrics are selected:

- Number of operations for creating and deleting component entities.
- Average delay time of message processing.
- Maximum line size.
- Average line size.
- Changing the amount of memory consumed.

4.2. Modeling context

Consider a system consisting of a single component – a message handler in public code repositories ($A1$) and a queue ($Q1$) that receives messages for processing (**Fig. 1**). Processing of these messages allows you to identify leaks

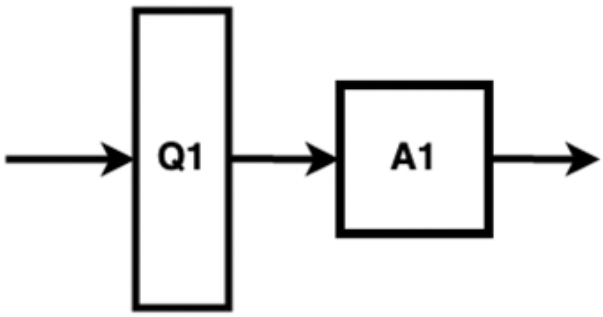

Fig. 1. *Processing system with queue.*

of source code or confidential information of companies.

During the simulation process, each algorithm receives information about the current load and returns – the number of component entities to be started or deleted at the next time.

Events on the github.com for the period from 2022-01-01 15:00:00 to 2022-01-12 02:51:46.

Additional modeling parameters are shown in **Table 1**.

Table 1

Modeling parameters

Parameter	Meaning
A1 performance	50 messages per second
Total number of messages	30 715 323
Time to start or delete entities	1 second

4.3. Modeling an algorithm with a threshold value

When modeling the algorithm with a threshold value, memory consumption was fixed.

As a result (**Table 2**) of the modeling, a minimal delay in message processing was achieved, but a huge number of operations for creating and deleting components.

Table 2

Results of modeling the algorithm with a threshold value

Parameter value	Number of operations
Number of scaling operations	81 746 850
Average queue size	3.534165161
Average message delay time	0.156226077
Maximum queue size	140

Table 3

Modeling results of the exponential smoothing algorithm

Parameter value	Number of operations
Number of scaling operations	148 122 776
Average queue size	10.47011898
Average message delay time	0.46282659
Maximum queue size	415

4.4. Modeling of the exponential smoothing algorithm

To model the algorithm, it is necessary to determine three constants:

- The effect of a smoothed series.
- Trend influence.
- Influence of seasonality.

However, before the load is formed and analyzed, it is impossible to correctly determine these parameters. The following values were randomly selected for modeling:

- The effect of the smoothed series – 0.4.
- Trend impact – 0.1.
- The influence of seasonality – 0.02.

As a result of the algorithm modeling, the memory consumption was fixed.

The algorithm showed results (**Table 3**) significantly worse than the algorithm with a threshold value. The number of operations is higher than 1.8 times, and the average delay is more than twice as compared to the algorithm with a threshold value.

4.5. Modeling a linear regression algorithm

When modeling the linear regression algorithm, memory consumption was fixed.

The algorithm showed a decrease in the number of operations for creating and deleting components by more than 740 times. However, message latency and queue size have increased significantly.

The result obtained (**Table 4**) can be explained by the fact that the algorithm

Table 4

Modeling results for linear regression

Parameter value	Number of operations
Number of scaling operations	109 596
Average queue size	7.624188911
Average message delay time	0.337023616
Maximum queue size	539

effectively smoothes the spikes and drops of the load to the average value and correctly determines trends.

4.6. Conclusions based on the modeling results

The algorithm with exponential smoothing is not applicable for this problem since it is necessary to strictly determine the initial constants to work correctly.

The linear regression algorithm showed a significant reduction in the number of component creation and deletion operations, compared to the classical threshold algorithm: 109,596 vs. 81,746,850 operations. However, the message delay time increases from 0.156 to 0.337 seconds.

The use of the linear regression algorithm allows to achieve a significant reduction in resource consumption with a minimal increase in delay time.

5. CONCLUSION

Using the linear regression algorithm for scaling components allows to significantly reduce the operations of creating and removing components (by more than 700 times), but the delay in message processing increases. This peculiarity can be leveled by adding a time window, on the basis of which the load forecasting will be performed.

By using the proposed algorithm for load prediction and the marking algorithm [11] for message routing in a stream processing system, it is possible to achieve a reduction of more than 800 times in the number of component scaling operations.

The proposed algorithms can be embedded in systems built on microservice [12] architecture. Thanks to the developed algorithms, resource consumption is significantly reduced, both when processing large amounts of data and when solving applied tasks.

REFERENCES

1. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025 // https://www.statista.com/statistics/871513/worldwide-data-created/ (accessed: 24.04.2023)
2. Loshin D. *ETL (Extract, Transform, Load): Business Intelligence*. USA, Silver Spring, Morgan Kaufmann Publ., 2012, 400 p.
3. Chandrima R, Kashyap B, Sandeep A, Manjusha P. Horizontal Scaling Enhancement for Optimized Big Data Processing: *Proceedings of IEMIS*. *Emerging Technologies in Data Mining and Information Security,* 2019, 639-649 p.p. DOI: 10.1007/978-981-13-1951-8_58.
4. Oancea B, Ciucu S. Time series forecasting using neural network. //https://arxiv.org/pdf/1401.1333.pdf (accessed: 24.04.2023)
5. Random Forests for Time Series // https://hal.science/hal-03129751/file/Block_bootstrap_for_random_forests.pdf (accessed: 24.04.2023)
6. Machine Learning Algorithms for Time Series Analysis and Forecasting //https://arxiv.org/abs/2211.14387 (accessed: 24.04.2023)
7. Gianluca B, Souhaib B, Yann-Aël B. Machine Learning Strategies for Time Series Forecasting. *Lecture Notes in Business Information Processing 138*, 2013, 62-77 p.p. DOI: 10.1007/978-3-642-36318-4_3.
8. Dastan H, Adnan M. A Review on Linear Regression Comprehensive in Machine

Learning. *Journal of Applied Science and Technology Trends,* 2020, 140-147 p.p. DOI: 10.38094/jastt1457.

9. Handanhal V. Forecasting With Exponential Smoothing – What's The Right Smoothing Constant? *Review of Business Information Systems,* 2013, 117-126 p.p. DOI:10.19030/rbis.v17i3.8001.
10. Chatfield C. The Holt-Winters Forecasting Procedure. *Journal of the Royal Statistical Society,* 1978, 264-279 p.p. DOI: 10.2307/2347162.
11. George G. Bulychev, Alexey V. Chernykh. Problems of Message Routing Algorithms in Streaming Data Processing Systems. *RENSIT: Radioelectronics. Nanosystems. Information Technologies,* 2022, 14(3):279-290e. DOI: 10.17725/rensit.2022.14.279.
12. Al-Debagy O, Martinek P. A Comparative Review of Microservices and Monolithic Architectures. *18th IEEE International Symposium on Computational Intelligence and Informatics*, 2018. DOI: 10.1109/CINTI.2018.8928192.